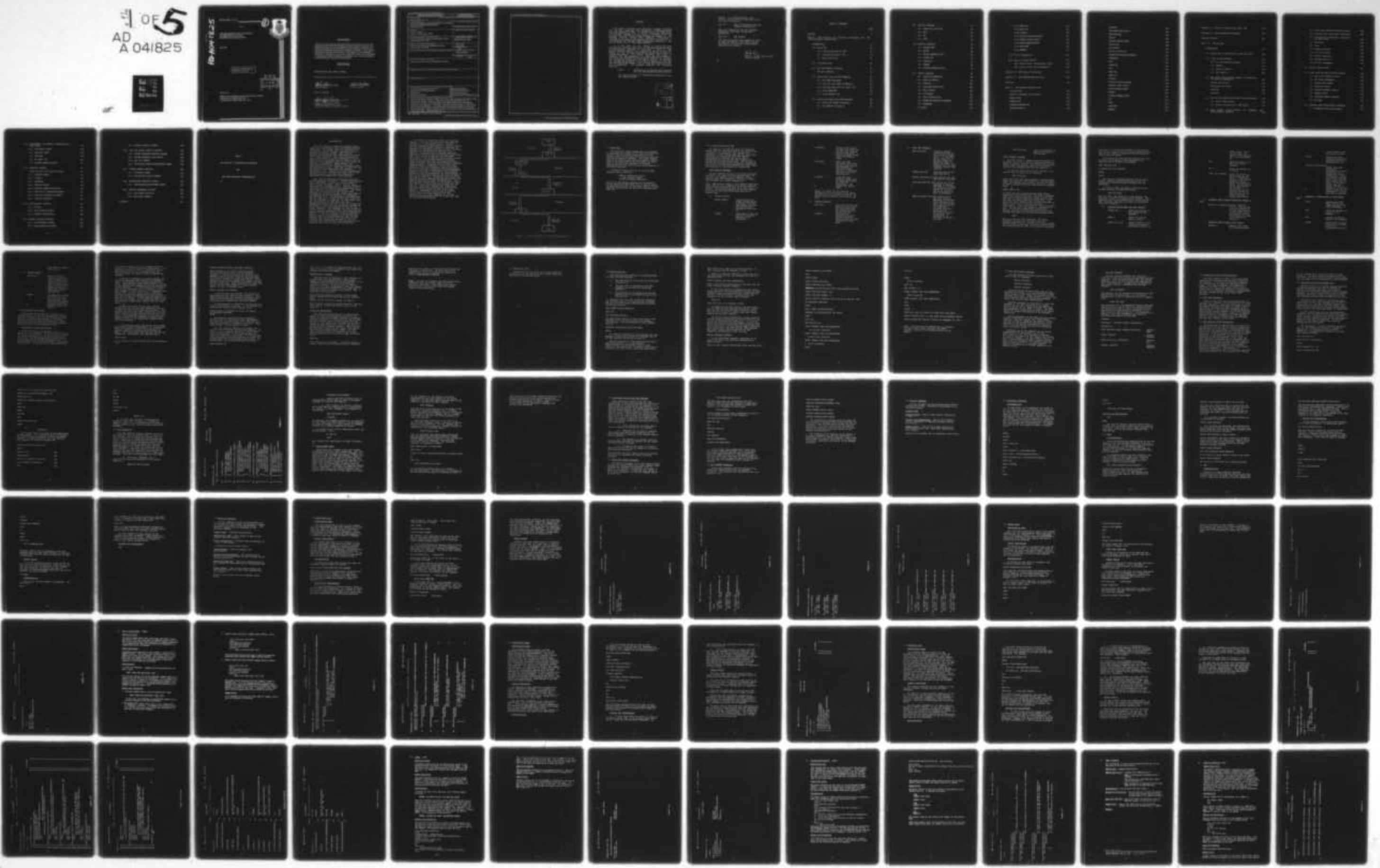
AD-A041 825 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370--ETC(U)
APR 75 C R MOORE, H J EIDEN

UNCLASSIFIED ESD-TR-75-88-VOL-3 NL

1 OF 5
AD
A041825



ESD-TR-75-88, Vol. III

1

B.S.

ELECTRONIC SYSTEMS DIVISION

USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370/158/OS/TSO INSTALLATIONS

April 1975

Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

LEGAL NOTICE

OTHER NOTICES

Do not return this copy. Retain or destroy.

"This technical report has been reviewed and is approved for publication."

Henry J. Eiden
HENRY J. EIDEN, Major, USAF
Program Manager

Charles R. Moore
CHARLES R. MOORE, GS-7
Project Engineer

FOR THE COMMANDER

Frank J. Emma
FRANK J. EMMA, Colonel, USAF
Director, Information Systems
Technology Applications Office
Deputy for Command & Management Systems

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER
ESD-TR-75-88, Vol. III

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370/158/OS/TSO INSTALLATIONS

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Charles R. Moore
Henry J. Eiden, Major, USAF

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Information Systems Technology Applications Office
Hanscom AFB, MA 01731

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
PE63101F/Project E189

11. CONTROLLING OFFICE NAME AND ADDRESS
Deputy for Command and Management Systems
Hanscom AFB, MA 01731

12. REPORT DATE
April 1975

13. NUMBER OF PAGES
415

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Aided Requirements Analysis
Information System Requirements
Requirements
Requirements Analysis
Requirements Analyzer
Requirements Specification
Logical System Analysis
Information System Development Tools

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The User Requirements Analyzer (URA) is one of two major components of computer aided requirements analysis. URA is an interactive software package that builds, and automatically analyzes information system requirements data bases for completeness, consistency, and ambiguity. In addition, it provides real-time update of, and keyword retrieval from the data base, and sophisticated text and graphics display of the target system requirements. URA is meant to be used in conjunction with the User Requirements Language described in ESD-TR-75-88, Vol. II.

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

PREFACE

This manual is based on the URA User's Manual prepared for the Air Force by the University of Michigan under contract F19628-74-C-0171. It differs from the University of Michigan product in so far as it concerns use of URA software with IBM OS/TSO system software as opposed to the Michigan Terminal System (MTS) software. The manual is intended to be used in conjunction with ESD TR 75-88, Vol I, "Introduction to Computer Aided Requirements Analysis", and ESD TR 75-88, Vol II, "URL USERS MANUAL".

It is the goal of this manual to assist the User Requirements Analyzer (URA) user in effectively manipulating the URA command language as specified in Part II "URA Command Descriptions". It is intended as an aid through all stages of the URL/URA requirements specification phase of system development. The manual specifies the steps in creating the URL database; i.e., inputting URL statements, modifying the contents of the database, generating outputs, and correcting syntactical and logical errors. It provides all information necessary to use URA in conjunction with the IBM 370/158/OS/TSO system. The original ESD version of URA, 2.0, is addressed. The document contains the following material:

Part 1 - URA/Version 2.0 Operating/Procedures for IBM 370/158/OS/TSO Installations

This part contains all information necessary to use the software under

OS/TSO. It is TSO-dependent, and requires modification by the user to fit his own installation.

Part II - User Requirements Analyzer Command Descriptions

This is a summary of the URA commands available, with their syntax and parameters.

Part III - URA Outputs

This part discusses the purpose and uses of each URA report, along with the way the reports fit into the requirements specification process.

Henry J. Eiden

HENRY J. EIDEN, Major, USAF
Program Manager

TABLE OF CONTENTS

PART I

URA VERSION 2.0 OPERATING PROCEDURES

FOR

IBM 370/158/OS/TSO INSTALLATIONS

Introduction

The first phase in using the URL/URA system deals with specifying a user requirement with URL statements. The second phase is concerned with using URA to enter the user requirement into a computer analyzable format. URA extracts information from these URL statements and stores it in a URA database. Once this information (a user requirement) is in the database, it can be modified, new information can be added to it, and reports can be generated presenting the status of the user requirements. These actions are implemented by the URA commands available in the URA (processing) mode. This mode of operation may be attained by accessing the URA software available in TSO. Therefore, by understanding the TSO commands that interact with URA, and the URA command language, the user can effectively manipulate the contents of a URA database.

This part of the manual specifies those TSO commands commonly used in interacting with URA and how to use them. It also serves as a guide to using the URA commands. Since there is such a large number of options and alternatives available in using each URA command, only those which are most useful to the user are presented. The reader should be made aware that both TSO and URA commands are specified in this manual and each set of commands can only be used in its respective (processing) mode. TSO commands can be used from time of signing on to TSO, to the time access to the URA software is acquired. At this point, only URA commands may be used until the user temporarily returns control to TSO (by hitting the ATTENTION or BREAK key), or terminates processing to be done in URA mode (through use of the URA "STOP" command). TSO commands can then be used up to the time of signing off TSO. This interaction between TSO and URA modes is better illustrated by Figure 1. To aid the reader in differentiating between TSO and URA commands, note that all URA commands will be indented farther than any TSO commands with which they may appear. At the terminal this indentation is not necessary and is provided here only for the sake of clarity.

The format of this part serves an important purpose. The first five sections deal with TSO and URA at an introductory level. Section 1 presents necessary information to the basic use of the IBM 370/158/OS/TSO. Section 2 explains the procedure of accessing URA once on TSO. Section 3 provides an introduction to the next five sections. Sections 4 and 5 present practical concepts and conventions to be known before using URA. Once access to URA has been achieved, Sections 6, 7 and 8 present the procedures needed to implement the various commands available. Several examples are given in these sections in order to better illustrate the results of specific implementations. Sections 9 and 10 deal with handling errors encountered in the use of URA. Appendix A of Part I presents several procedures to aid in the task of modifying the URA database. Appendix B presents a list of all URA commands available (and the parameters for each command) as well as the abbreviations for all these to serve as a quick reference. Throughout this manual the long form of URA commands and parameters are used interchangeably with their abbreviations. The sections are ordered then, in the manner in which the user will be using the two systems (TSO and URA) and from basic introductory material to specific information dealing with a particular aspect of usage (using the TSO editor, for example). Therefore, as the user of TSO/URA becomes more and more familiar with the implementation of those systems, the latter sections and appendices will be in more use than the introductory sections.

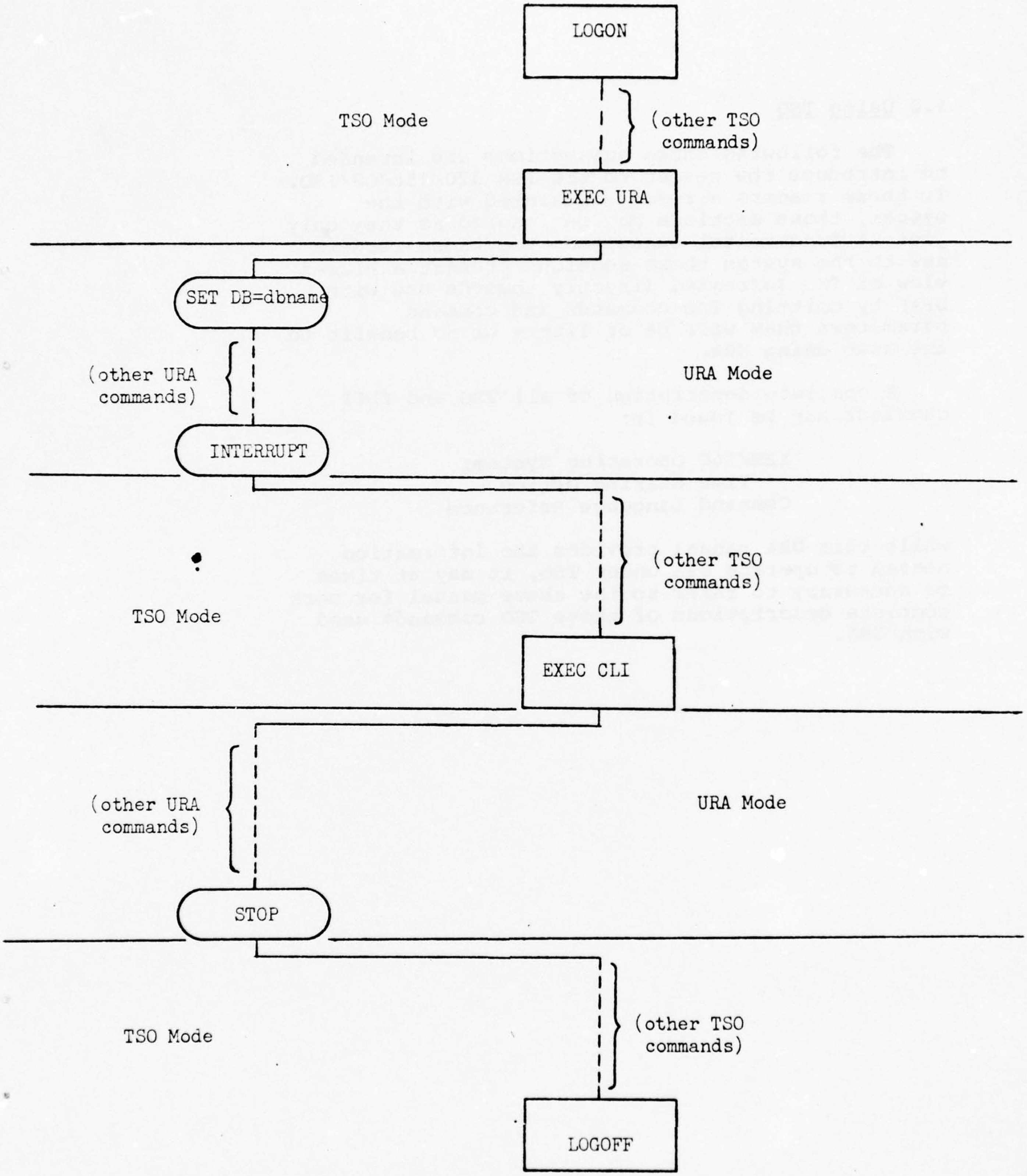


FIGURE 1. Interaction between TSO and URA processing modes

1.0 Using TSO

The following three subsections are intended to introduce the reader to the IBM 370/158/OS/TSO. To those readers already acquainted with the system, these sections may be ignored as they only present fundamental concepts. For those readers new to the system these sections present a biased view of TSO (oriented directly towards use with URA) by omitting TSO commands and command parameters that will be of little or no benefit to the user using URA.

A complete description of all TSO and EDIT commands may be found in:

IBM/360 Operating System:
Time Sharing Option
Command Language Reference

While this URA manual provides the information needed to operate URA under TSO, it may at times be necessary to refer to the above manual for more complete descriptions of those TSO commands used with URA.

1.1 A Brief Overview of TSO

In general, the URA user will be concerned either with the running of URA or the preparation of datasets as inputs for the various URA commands. In the former case, URA is called and initiated by issuing the EXEC URA command. Preparation of datasets for use in URA is accomplished by issuing the EDIT command and giving the desired dataset name. The concern of the following sections is to provide information relative to the TSO EDIT commands and other basic TSO commands as they relate to running URA.

TSO Command Language

In any terminal session it is first necessary to LOGON. Once this has been accomplished, the user has the computer at his command. This section lists the commands in their most common form followed by a brief explanation.

NOTE: All dataset names in this section have been given dataset type ".DATA" as the great majority of user defined datasets will be of this type. In general, user defined dataset names will consist of an up to eight character user name and a dataset type.

A. Global Control

LOGON TSXXXX — tells TSO that ID number TSXXXX wishes to LOGON. TSO will then prompt the user for the password associated with the ID.

LOGOFF — tells TSO to sign the user off and print a summary of run statistics.

SIZE(XXX) Controls the amount of space the user has in core (for running URA the user must enter size(246) which gives him 246K of core).

PROC(XXX) Tells TSO that procedure "XXX" will be used (for running URA PROC(URA) is specified which allocates sufficient datasets to run all URA reports).

TERM(XX) tells TSO that terminal number "XX" is being used (only for IBM 2741 terminals). Any number 1-99 may be used.

NOTE: If either of the last two statements are not given in user LOGON, the user will be prompted for them. No prompt will be given if the size is not set.

B. <u>Program Control</u>

EXEC XXX tells TSO to load and execute program XXX. The user will most frequently use EXEC URA or EXEC CLI.

FREEALL releases all allocated datasets (should be given after issuing an ATTN Interrupt to exit URA).

C. Data Set Handling

EDIT XXX.DATA — creates a dataset named "XXX.DATA" if one does not already exist. If one does exist, tells TSO to put the editor in charge in order to modify XXX.DATA. A complete list of EDIT commands follows this section.

DELETE XXX.DATA — destroys the dataset named XXX.DATA.

RENAME XXX.DATA YYY.DATA — changes the name XXX.DATA to YYY.DATA.

COPY XXX.DATA YYY.DATA — copies all of the contents of dataset XXX.DATA to YYY.DATA. Any information which exists in the data set being copied to will be destroyed.

MERGE XXX.DATA YYY.DATA — copies all or parts of dataset XXX.DATA to YYY.DATA. Line numbers may be given to state which line numbers are to be taken from XXX.DATA and after which line in YYY.DATA they are to be written. No information in YYY.DATA is changed or lost.

LIST XXX.DATA	lists the contents of dataset XXX.DATA at the terminal.

EDIT Command Language

The TSO editor is used to create datasets to be input to URA or to modify existing datasets in order to correct errors. The editor thus has two modes: INPUT for putting lines into the dataset and EDIT for changing lines in the dataset.

If the user wished to create a dataset to be used by URA, he will use the command:

```
EDIT URA.DATA
```

This line tells the computer that a dataset named "URA" with dataset type ".DATA" is to be editted. Since the dataset is a new one, TSO will be unable to find a dataset with this name and will respond with:

```
ENTER NEW or OLD
```

to which the user should respond "NEW" before hitting the carriage return. The "NEW" statement tells the computer that the dataset is a new one and that it should respond in INPUT mode so that the user may type lines into the dataset. TSO will then print INPUT at the terminal followed by 00010 and wait for the user to provide information. After each carriage return, TSO will print out the next line number in increments of ten. If the user inadvertently specifies a dataset name which he has already used and uses the NEW statement the computer will respond with:

```
EDIT
```

and will wait for EDIT commands. The above procedure assures that the user will not inadvertently destroy a dataset by creating a new dataset with an old name. The user should receive the ENTER NEW OR OLD line if the name is actually a new one. If it is not received (in which case EDIT will be printed out), the user should realize that he has chosen a previously defined name and should choose another.

If the user is sure that the dataset name has not been used before, he may abbreviate the creation of the dataset by entering:

```
EDIT URA.DATA NEW
```

to which TSO will respond:

```
INPUT

00010
```

The user is cautioned however that if he has chosen a previously defined dataset name, the information contained in that dataset will be destroyed.

If a user wishes to change a dataset he has already created, he need only enter:

```
        EDIT XXX.DATA
```

TSO will then type EDIT out at the terminal and wait for the user to specify EDIT commands. The following give a brief description of the various EDIT commands (synonyms are given in parenthesis after the command).

A. <u>Commands Which Move the Line Pointer</u>

BOTTOM (B) — moves the pointer to the bottom of the dataset.

DOWN n1 — moves the pointer down n1 lines.

LIST (L) n1 n2 — causes editor to list lines n1 through n2 and causes pointer to point to n2. If n2 is not given, the pointer will point to and list line number nl.

TOP — moves the pointer to the top of the dataset.

UP n1 — moves the pointer up n1 lines.

FIND (F) "STRING" — cause the editor to search line by line until it finds the string. Line pointer points to line in which the string is found. If the string is not found "TEXT NOT FOUND", is printed out and the pointer returns to the position it had before the FIND command was issued.

B. Commands Which Change Characters Within a Line

CHANGE n1 /STRING1/STRING2/ replaces string 1 with string 2 in line number n1. One may also specify a line number range and specify that all occurrences of string 1 be replaced by string 2.

C. Commands Which Change Entire Lines

DELETE n1 — deletes line number n1 from the dataset.

A line number range may also be given.

INSERT — inserts the data provided immediately following the line being pointed to.

Insert/Replace/Delete — allows insertion, deletion or replacement of lines by typing a line number and the new information. If the line number already exists, the new line will replace the old line. If the line number does not exist, one will be created. If the line number exists and no information is given, the line is deleted.

D. Commands to Change Mode or Leave EDIT Mode

INPUT — switches to INPUT Mode, starts line numbering after last line of dataset.

SAVE — saves all changes and inputs to the dataset.

END — returns to TSO for further TSO commands.

RETURN — hitting the carriage return with no input line will automatically switch from one EDIT mode to the other.

E. Miscellaneous

VERIFY (V) with the CHANGE Command, causes the line in which the change has been made to be printed out. With the FIND command causes the line in which the string has been found to be printed out.

RENUM renumbers the dataset in intervals of 10. User may also give range within a data set to be renumbered and may specify any integer interval he wishes.

1.2 Interactive Use of TSO

Interactive use of TSO will commonly be through the use of IBM 2741 Terminals or smaller portable terminals such as the CDI Teleterm 300. The use of either type terminal is virtually identical after the user has logged on. There are, however, some differences in the initiation sequence which will be mentioned below.

Initiating a Terminal Session

Before dialing up the computer, it is necessary that whichever terminal is being used be set up to talk with TSO. If an IBM 2741 is being used with a dataphone, it is only necessary to assure that the two switches on the left side of the terminal be set to COM and AUTOEOT OFF. If these switches are not set properly, no connection will be made. If an IBM 2741 is being used with an accoustic coupler, it is necessary that the coupler be turned ON and set to HALF-DUPLEX. If a portable terminal is used, it should be set to HALF-DUPLEX and 30 cps. In every case, the terminal should be turned on before dialing the computer.

In order to initiate a terminal session, the user must then dial into TSO via the telephone lines. Infrequently, the phone lines will be busy. This generally means either that all lines to the computer are temporarily filled or that TSO is not operating. If a busy signal is received, hang up and call again in a few minutes. If a busy signal is still received, it is probable that TSO is down. In the latter case, a recorded message is sometimes placed on the line saying that TSO is down and giving some indication as to when it will return. If the computer number rings but does not answer, it is likely that TSO is in the process of going down, hang up and try again later.

Assuming the lines are not busy and the computer is up, the user will generally hear one or two rings and then a high pitched tone. If an IBM 2741 is being used with a dataphone, the user should depress the DATA button when he hears the tone. If an accoustic coupler is being used with either an IBM 2741 or a portable terminal, the handset should be fit snugly into coupler after the tone is heard.

If all has gone well, TSO should then respond with "WHICH SYSTEM DO YOU WANT". If It does not, there is probably a switch set wrong and all switches should be checked before dialing in again. The user should then type TSO and hit the RETURN key. After this, TSO responds with:

```
ENTER LOGON
```

At this point, the user should enter the following line:

```
LOGON TSXXXX SIZE(246) PROC(URA) TERM(XX).
```

where TSXXXX is the user's account number, size(246) reserves 246K of core, PROC(URA) allocates a sufficient number of datasets, and TERM(XX) tells TSO which terminal is being used. If the user neglects to enter the PROC and TERM statements, he will be prompted for them. If the terminal number is not known, any number between 1 and 99 may be used. After entering this line, TSO checks to make sure TSXXXX is a valid account number, if so, it responds with:

```
ENTER PASSWORD FOR TSXXXX.
```

It then returns the carriage and prints out some characters to mask the user password. The terminal will make three passes, filling up eight spaces with characters and then wait while the user types his password and then hits RETURN.

If the password is incorrect, the user will be told so and asked to re-enter. If the password is correct, the terminal will usually print:

```
TSXXXX LOGON IN PROGRESS AT (time) ON (date)
NO BROADCAST MESSAGES
READY
```

Occasionally, there will be a broadcast message (such as "TSO WILL BE DOWN NOON HOUR"), and there may be some delays after entering the password and before the READY is received. These occur mainly when the system is busy.

Once the terminal responds with READY, the user is able to execute TSO and URA commands. If a user is on a terminal whose line size is less than 120 characters, it is necessary that he inform the computer of this fact. If he does not upon execution of any URA commands, an error will be generated and the URA program will abort. To prevent this, the user should enter the line:

```
TERM LINESIZE XX
```

where XX is the number of characters per line for the terminal and then hit RETURN, after which the computer responds with READY.

Terminating a Session

Once the user is finished at the terminal, it is necessary that he sign-off. This is done simply by typing LOGOFF and hitting RETURN. It is necessary that the user be in TSO when this command is issued. If he is still in URA, an error message will be generated. The user should then enter the STOP command of URA, wait until he receives a READY, and then enter LOGOFF. The terminal will then print:

```
XXXX CORE USED XXTGETS XX TPUTS XX DISC EXCPS
JOBLOG XXXXXXXXXXXX X.XXCPU SECS XXX.XX ELAPSED
SECS
TSXXXX LOGGED OFF TSO AT (time) ON (date) +
```

where the X's represent various numbers. TSO will then hang up and the user should turn off the terminal.

Attention Interrupts

The ATTN (or BREAK) key may be used anytime after the user has successfully logged on and before he enters the LOGOFF command, to interrupt whatever processing is being done by TSO (or URA) at the time and return to the terminal for TSO commands. One common use is to get out of URA to create or edit files to be used by URA later on during the same terminal session. After hitting the ATTN key, there will be a short delay until the terminal responds with a READY. At this point, any necessary TSO commands may be run. When the user wishes to return to URA, he should enter:

```
EXEC CLI
```

which will act as a restart. The only caution is against using an ATTN interrupt while a database modification (adding or removing data from a URA database) is being run. In these cases, the possibility exists of causing the database to become irrecoverably unusable.

NOTE: For sake of economy, the user should enter a FREEALL upon receiving a READY after an Attention Interrupt. This step is not always necessary but will make the EXEC CLI restart execute more efficiently.

1.3 Batch Use of OS

Information on the batch use of URA OS will be provided as an attachment to this manual when this information becomes available.

2.0 Initiating URA

The preliminaries involved in initiating URA are very straight forward:

a. The URA database file must be created and initialized.

b. URA mode must be entered so that URA commands may be used to access the database.

c. The database to be accessed by the URA commands must be assigned by way of the SET command.

Assuming that the user is already logged on TSO and the name of the database will be URADB, the previous three steps are carried out by the following statement:

```
EXEC NEWDB 'uradb.database'

EXEC URA

SET DB=uradb.database
```

The EXEC NEWDB command is only given when a new database is to be created and is not needed subsequently. The TSO response to this command will be:

```
DATABASE INITIALIZED WITH 150 PAGES

READY
```

The EXEC URA command is used whenever the user wishes to enter URA in order to implement any of the URA commands specified in Part II.

The SET command is a URA command which tells URA which database is to be accessed. URADB.DATABASE is the default value if this command is not specified. If another database is specified, it will be used until the user exits

URA (either by a STOP or an ATTN interrupt), or until another SET DB command is given.

After the EXEC URA command is given and after the completion of each URA command, the user is prompted with:

```
ENTER COMMAND (AND ANY PARAMETERS),
```

after which the carriage returns and waits for the user to enter the next command.

If the user wishes to temporarily exit URA to return later in the same terminal session, an ATTN interrupt may be given which will return the user to TSO. After the user is finished with TSO, commands, he may re-enter URA by typing:

```
    EXEC CLI
```

which will act as a restart to URA.

To terminate the URA session, the user should use the STOP command (not the interrupt), which will return him to TSO. Once in TSO, he may do whatever TSO commands he wishes and then logoff, ending his terminal session.

The final section represents a sample terminal session roughly following the diagram found at the end of the introduction (Figure 1). It is to be noted that while the session appears to move along swiftly in written form, there will be delays when it is being run. The user should expect delays of some length after the EXEC URA and STOP commands and lesser delays after other TSO and URA commands, particularly if TSO is busy.

Sample Terminal Session

In the following, computer responses are in capital letters, user provided information in small letters.

(Dial up TSO, depress DATA button upon hearing tone)

```
WHICH SYSTEM DO YOU WANT?

tso

ENTER LOGON

logon tso593 size(246)

ENTER PASSWORD FOR TSO593

■■■■■■■■ (password typed over mask provided by TSO)

ENTER PROCEDURE NAME - ura

ENTER TERMINAL NUMBER - 9

TSO593 LOGON IN PROGRESS AT 11:44:32 on FEB 29, 1975

NO BROADCAST MESSAGES

READY

exec newdb 'uradb.database'

DATABASE INITIALIZED WITH 150 PAGES

READY

exec ura

URA VERSION 2.0

ENTER COMMAND (AND ANY PARAMETERS)

     set db=ura. database

ENTER COMMAND (AND ANY PARAMETERS)

     } (other URA commands)

ENTER COMMAND (AND ANY PARAMETERS)

!  (ATTN interrupt)

READY
```

```
FREEALL


READY
    }(TSO commands)
exec cli

URA VERSION 2.0

ENTER COMMAND (AND ANY PARAMETERS)
     }(URA commands)
ENTER COMMAND (AND ANY PARAMETERS)

stop

READY

logoff

200K CORE USED 122 TGETS 150 TPUTS 648 DISK EXCPS

JOBLOG 750594272220 11.3 CPU SECS 600.48 ELAPSED SECONDS

TSO593 LOGGED OFF TSO AT 11:55:20 ON FEBRUARY 29, 1975
```

NOTE: The LOGON may be abbreviated by entering the line logon ts0593 size(246) proc(URA) term(97), instead of the LOGON entered in the sample.

3.0 The URA Command Language

The URA Command Language consists of three basic types of commands:

Report Commands

Modifier Commands

Control Commands

Report commands retrieve data from the URA database and output it in some meaningful format. These reports do not change the contents of the database whatsoever. Their purpose is solely that of displaying orderings and/or relationships within the current user requirements.

Modifier Commands are intended to modify the contents of the URA data base in the manner defined by the user. These commands take legal URL statements or URL names as input. URA then generates error diagnostics as well as an output report to present the outcome of the database alteration.

Control Commands are the means to pass certain control information to the User Requirements Analyzer. The SET command, for example, allows the user to define which URA database is to be accessed by the Report and Modifier Commands as well as setting various switches and assigning input and output files.

Although any of the commands can be issued independently of each other, it is often advantageous to use some commands in sequence. This means that output of one command can be used as input by another. The most common instance of this is when NAME-GEN is used to select certain names (say all PROCESSES for example) which can then be used as input to a Report Command (possibly PICTURE, for a PICTURE REPORT for all PROCESS names).

The HELP Command

The HELP command provides the user with information about the syntax and parameters of URA commands. When the user enters HELP, the system displays a list of all available URA commands and their abbreviations. By specifying a particular URA command name as a parameter to the HELP command:

```
HELP CONTENTS
```

for example, all the parameters available for this command will be printed out. If the "LONG" parameter was given in conjunction with a command name:

```
HELP FPS LONG
```

all parameters for the FORMATTED-PROBLEM-STATEMENT would be printed out as well as a description of the function of each of these parameters for the command. This description is presented in the same format as that in Part II. To illustrate an example, when "HELP CONTENTS" was given the following information was printed:

```
CONTENTS

Prototype:  CONTENTS (CONT) (parameter)...

Parameters:

FILE (dataset name), NAME(N)=user-name      Default:
                                            FILE

INDEX, NOINDEX                              Default:
                                            NOINDEX

LEVELS=integer, LEVELS=ALL                  Default:
                                            ALL

NCFLAG, NONCFLAG                            Default:
                                            NONCFLAG
```

4.0 Specifying Input to URA Commands

For most commands in URA, one or more names (specified by the user) can be used as input to the command. In the case of Modifier Commands, the modification is made for each name used as input. For Report Commands, information is retrieved for each of the names used as input. Except for the INPUT-URL command, all names used as input to the Modifier and Report Commands must be names already stored in the user's URA database.

4.1 The NAME Parameter

There are two methods of specifying names to be input to a command. The simplest way is to use the NAME parameter. When this parameter is used, the modification will be made, or report will be generated, for only that name specified by the NAME parameter. For example, if NAME=T-CARD were used for the DELETE command, only T-CARD would be deleted from the data base. Likewise, if NAME=T-CARD were used as a parameter for the CONTENTS command, the CONTENTS REPORT would be generated for the name T-CARD, and no others.

4.2 The FILE and INPUT Parameters

The second way to specify names as input to a URA command is to put all the names for which the modification is to be made, or report generated, into a file and specify that the contents of that file are to be used as input via the FILE or INPUT parameter. FILE and INPUT are different in the way names can be formatted within the file specified by these parameters. When using the FILE parameter, each name in the specified file must begin in the first column, of each line, of the file and only one name per line is allowed. The format for files specified by the INPUT parameter varies accordingly to the URA command using this parameter. For example, if INPUT=IFILE.DATA were used as a parameter for the INPUT-URL command, the file, IFILE, must consist of URL statements to be entered into the URA database. For those Modifier Commands that use the INPUT parameter, the particular format needed for the input file is specified in later sections.

4.3 Entering Data Into An Input File

Before specifying a dataset name via the FILE or INPUT parameter, one must go through the process of entering data into the dataset according to the format required by the particular command. In order to enter lines into a dataset, the user must first access the dataset via the EDIT command and then use the INPUT mode of EDIT.

If the user is creating a new dataset (see last section), TSO will automatically respond in INPUT mode. If the user is adding lines to an already existing dataset, the user should enter EDIT mode and then hit the carriage return again causing TSO to automatically switch to INPUT mode.

Once in INPUT mode, TSO will automatically increment the line number by 10 with each carriage return, until a blank line is entered at which point TSO will switch to EDIT and the user should issue the SAVE and END commands. If the SAVE command is not given, none of the provided information will be saved.

The procedure to create a dataset, copy data into it and destroy the dataset, is shown in Example 4a. All uppercase lines are TSO responses, all lowercase are typed by the user. (R) signifies RETURN.

```
edit ura.data (R)

ENTER NEW OR OLD-new (R)

INPUT

00010 process p1; (R)

00020 description; (R)
```

```
00030 this is a process to be used (R)

00040 in my problem statement; (R)

00050 eof (R)

00060 (R) (denotes blank line return)

EDIT

save (R)

SAVED

end (R)

END

delete ura.data (R)

READY
```

Example 4a

As in most cases, the above can be abbreviated to some extent. It is also possible to eliminate the automatic line numbering by using the NONUM command. A similar example, this time without line numbers, is found in Example 4b.

```
edit ura.data new nonum (R)

INPUT

process p1;                             (R)

description:                            (R)

this is a process to be used            (R)

in my problem statement;                (R)

eof                                     (R)
```

```
(R)

EDIT

se (R)

SAVED

READY

d ura.data (R)

READY
```

Example 4b

It is noted that the EOF is necessary for datasets being used as inputs to a INPUT-URL command. Also, SE is an abbreviation for the two commands SAVE and END, (D is an abbreviation for DELETE).

4.4 Using NAME-GEN

Many URA commands allow or require a list of user defined names of various types as inputs to the command. One alternative to typing in long lists of names is to allow NAME-GEN to enter the names. The various parameters for NAME-GEN (NG) allow the user to select particular types of names (such as all the GROUPS and ENTITIES) and have NAME-GEN list all of these names in a dataset named URANAMES. The names are formatted one per line starting in the first column of the line as required by other URA commands. The contents of URANAMES may then be used as input to a URA report command. This may be accomplished in three ways:

1. The value, URANAMES, can be specified for the FILE parameter in the Report Commands:

```
NAME-GEN ENTITY GROUP
```

```
URA VERSION 740710                                              JUL 28, 1974   21:34.54       AGE
                                       ADS-EXAMPLE

                                     CONTENTS REPORT

PARAMETERS FOR: CONT

   FILE NONCFLAG NOINDEX LEVELS=ALL


   1*     1 CHECK (GROUP)

   2*     1 EMPLOYEE-NAME (GROUP)
    1      2    FIRST-NAME (ELEMENT)
    2      2    LAST-NAME (ELEMENT)
    3      2    MIDDLE-INITIAL (ELEMENT)

   3*     1 EXPLICIT-PERSONAL-DATA (GROUP)

   4*     1 FIXED-EMPLOYEE-DATA (ENTITY)
    1      2    EXPLICIT-PERSONAL-DATA (GROUP)
    2      2    EMPLOYEE-NUMBER (ELEMENT)
    3      2    EMPLUYEE-NAME (GROUP)
    4       3      FIRST-NAME (ELEMENT)
    5       3      LAST-NAME (ELEMENT)
    6       3      MIDDLE-INITIAL (ELEMENT)
    7      2    RELATED-PERSONAL-DATA (GROUP)

   5*     1 RELATED-PERSONAL-DATA (GROUP)

   6*     1 TERMINATED-EMPLOYEE-PART (GROUP)
    1      2    EMPLOYEE-NAME (GROUP)
    2       3      FIRST-NAME (ELEMENT)
    3       3      LAST-NAME (ELEMENT)
    4       3      MIDDLE-INITIAL (ELEMENT)
    5      2    TERMINATION-CODE (ELEMENT)
    6      2    EMPLOYEF-NUMBER (ELEMENT)

   7*     1 VARYING-EMPLOYEE-DATA (ENTITY)
    1      2    PAYRATE (ELEMENT)
    2      2    EMPLOYEE-NUMBER (ELEMENT)
    3      2    TAX-RATE (ELEMENT)
    4      2    NUMBER-OF-DEPENDENTS (ELEMENT)
    5      2    YTD-DEDUCT (ELEMENT)
    6      2    YTD-GROSS (ELEMENT)
```

Example 4.c

```
CONTENTS FILE=URANAMES
```

2. Specify the FILE parameter with no corresponding value, when no value is assigned, URANAMES is taken as the default.

3. Most simply, not specify anything. If neither NAME nor FILE are used as parameters, the default is again URANAMES. For example, by specifying:

```
NAME-GEN ENTITY GROUP

CONTENTS
```

in sequence, the CONTENTS REPORT will be generated for all ENTITY and GROUP names in the URA database as shown in Example 4c. The contents of URANAMES remains until another NAME-GEN is issued.

The above may be further abbreviated using URA abbreviations. Thus:

```
NG ENT GR

CONT
```

will provide the same output as found in Example 4c.

4.5 Using PUNCH Files

PUNCH files are datasets which have formats acceptable by FILE or INPUT parameters. The file described in the previous section, URANAMES, is a PUNCH file from the NAME-GEN command. Output from NAME-GEN is put into its assigned PUNCH file so that it may be used as input to any of the FILE parameters for Modifier and Report Commands. The PUNCH file format is different from the report format for the command that generates both of them. For example, implement the NAME-GEN command for all PROCESSES. The report generated for this request will consist of a report heading, line numbers for the contents of the report, the names of all PROCESSES in the database and their corresponding name type (which is, of course, PROCESS). If the user copies the contents of the PUNCH file produced by this command:

```
LIST URANAMES
```

all that will be in this file is the names of the PROCESSES, no report heading, etc. In other words, the PUNCH file contains similar information to the report from the command, but in a format acceptable to the FILE and INPUT parameters of other URA commands.

NAME-GEN PUNCH output does not have to be written into URANAMES. This is merely the default PUNCH file for NAME-GEN. For any command that utilizes the PUNCH parameter, a PUNCH file may be specified in the following manner:

```
PUNCH=dataset name
```

This is most often done when PUNCH information from the FORMATTED-PROBLEM-STATEMENT or PRINT-COMMENT-ENTRY command is desired. The information in the PUNCH file might then be edited and reentered into the database. The following sequence of commands is common:

```
PCOM PUNCH=PUNCH N=F-DATA DESC

EDIT PUNCH

CHANGE 20 /will change less/cannot be changed more/

SE

EXEC URA

    RCOM INPUT=PUNCH N=F-DATA
```

The DESCRIPTION comment entry for F-DATA is written into PUNCH. Minor editing is performed on the DESCRIPTION and then the contents of PUNCH is used as input the the RCOM command so that the old DESCRIPTION comment entry is replaced by the modified comment entry in PUNCH. Specific usage of the PUNCH parameter is given in the descriptions of the individual commands that utilize this parameter.

5.0 Receiving Output From URA Commands

When generating outputs from URA, the information is put into a dataset or printed on a device such as a line printer or terminal. If this dataset or device is not specified (i.e., no dataset is assigned to the OUTPUT parameters of the URA "SET" command) then all outputs are written to the terminal which is the main output file or device. This means that output will be written on the terminal when in conversational mode and on the line printer when running batch. There are several reasons why you would want to route the outputs elsewhere, especially if in on-line processing:

a. Large quantities of output would take too long to be printed at the terminal.

b. Depending on the type of terminal used, some portions of the output may not be printed because of physical restrictions imposed by the terminal.

c. The handling of printout from the terminal can sometimes be awkward and the format not asthetically pleasing.

d. No copy of the output is desired. (Only the PUNCH file may be needed as a step in a modification procedure).

The following sections present some alternatives available in receiving the output from URA commands.

5.1 Using the OUTPUT Parameter

The OUTPUT parameter in the SET command allows the user to specify where all output (except PUNCH information) generated by commands is to be printed. If nothing is specified, all output is sent to the terminal. To specify that output is to be sent elsewhere, the URA SET command should be given as:

```
          SET OUTPUT=dataset-name
```

URA will then copy all information into the dataset which may then be listed on a high speed printer. If the user then wishes to return the output to the terminal, he may issue:

```
          SET OUTPUT=*
```

A good example of how these parameters are used is the following sequence of URA commands.

```
SET OUTPUT=OUTPUT.DATA

NAME-GEN ALL

FPS

NAME-GEN PROCESS

PICTURE

SET OUTPUT=*

NAME-GEN UNDEFINED

CHANGE-TYPE TYPE=GROUP
```

In the above, four reports were copied into the dataset named OUTPUT.DATA as it is probable that these printouts will be fairly large and would be better printed on a high speed printer. The output was then sent back to the terminal (*) so that some editing might be done on the database (in this case all undefined names changed to GROUPS).

5.2 The_NOPRINT_Parameter

Several URA commands allow the option of not having the output printed via the NOPRINT parameter. The commands that currently allow this parameter are:

DELFTE-COMMENT-ENTRY (DCOM)

FORMATTED-PROBLEM-STATEMENT (FPS)

NAME-GEN (NG)

PRINT-COMMENT-ENTRY (PCOM)

PROCESS-INPUT-OUTPUT (PRIO)

REPLACE-COMMENT-ENTRY (RCOM)

The two Modifier Commands RCOM and PCOM have this parameter available because the printout can be fairly large and may not be needed for future reference. The report heading for the RCOM or PCOM output, and any error diagnostics are still printed to provide a hard-copy record of the command implementation.

The remaining four commands (which happen to be Report Commands) can use this parameter in conjunction with the PUNCH parameter. When PUNCH information is desired, there may be no need for the report, therefore, the option of the NOPRINT parameter.

6.0 Control Commands

All the commands in this section relay control information to URA. Each will be described in the following format:

Command name

Implementation - how to send control information to URA.

Options and Alternatives - how to use different command parameters to specify different control information.

Common Errors - some of the common logical and syntactical errors that occur when implementing this command.

Examples are provided for the commands listed below.

6.1 Attention Interrupt

Implementation

In many cases, it is convenient to return to TSO to create and copy information into datasets or edit existing datasets. When this is the case, the user may get out of URA by hitting the ATTN or BREAK key after which TSO will first type an exclamation mark (!) and after a brief delay will type READY. After all TSO commands are completed, it is necessary to issue an EXEC CLI to return to URA (an EXEC URA may also be given but this will take much longer and is unnecessary). An example of using TSO in this manner is shown in Example 6.1a.

```
!

READY

freeall

READY

edit i.data new

INPUT

00010 process:  field-check-new;

00020 part:  new-employee-updating;

00030 utilized by:  new-infor-validation;

00040 eof

00050 (RETURN)

EDIT

se

SAVED
```

```
READY

EXEC CLI

     INPUT-URL I=I.DATA UPDATE
```

Options and Alternatives

None

Note

For sake of economy, the user should enter a FREEALL upon receiving a READY after an Attention Interrupt. This step is not always necessary but will make the EXEC CLI restart execute more efficiently.

6.2 SAVE

Implementation

Before performing any modifications on the URA database (via the Modifier Commands) it may be desirable to save the contents of the URA database as a precaution. If TSO crashes during a modification procedure it is possible that the contents of the URA database being modified will become unusable.

To safeguard against any such disaster, the user may wish to make a copy of his database before attempting any modification. This is readily accomplished in TSO using the COPY command. The user should issue:

```
COPY URADB.DATABASE SAVE.DATABASE
```

which will cause TSO to create a compatible database file and copy the contents of URADB.DATABASE into it. If the dataset being copied into already exists, the user will be prompted with:

```
DATASET SAVE.DATABASE IS ABOUT TO BE REUSED
```

and will then allow the user to either give the command the go ahead (at which time all previous contents of SAVE.DATABASE will be lost) or respecify the command. If the dataset with the name does not exist, one will automatically be created.

If no disaster occurs, the SAVE.DATABASE may be destroyed by issuing

```
DELETE SAVE.DATABASE
```

If a disaster has occurred, the contents may be recalled in either of two ways. The first and least expensive is to use the RENAME command after deleting the unusable database:

```
RENAME SAVE.DATABASE URADB.DATABASE
```

Using this method, the user returns the database to the contents it was before the disaster but no longer has a copy of his database. The user may also use the COPY command after the unusable database has been deleted:

```
DELETE URADB.DATABASE

COPY SAVE.DATABASE URADB.DATABASE
```

If the user no longer wishes a copy he may issue:

```
DELETE SAVE.DATABASE
```

and he will be left with his original database.

6.3 SET

Implementation

There are so many different possible implementations of this command that only the more valuable ones will be shown. The most common form of this command is using OUTPUT and DB parameters:

```
SET DB=URADB.DATABASE OUTPUT=OUTPUT.DATA
```

The DB parameter specifies the database file to be accessed by subsequent URA commands. OUTPUT sends all report data to the specified dataset. These parameters retain their values until the parameters are reassigned (via another SET command) or the URA run is terminated by the STOP command or an Attention Interrupt.

Options and Alternatives

Another parameter which might prove valuable is the INPUT parameter. By specifying INPUT:

```
SET INPUT=COMMANDS.DATA
```

URA will use the dataset for a source of URA commands. This is desirable when a certain sequence of commands are often repeated or when the system is slow and the user wishes to spend time elsewhere. This could be done as follows:

```
!

READY

FREEALL

READY


EDIT commands.data nonum new

input

set db=uradb.database

ng all

fps

dictionary
```

```
ng prc

picture

(blank line RETURN)

EDIT

se

SAVED

READY

EXEC CLI

    SET I=COMMANDS.DATA
```

URA will execute all the commands in the file COMMANDS.DATA and then return control to the user. COMMANDS.DATA could be kept for later use if this is desirable.

Common Errors

It is very important to keep track of how you have set the various parameters. Most of all, be sure that you have set DB equal to the database you want accessed. If this is not set, it defaults to DB=URADB.DATABASE which may not be the database you want to access.

6.4 STOP

Implementation

This is the easiest command to implement. By specifying:

```
STOP
```

you terminate the URA run and return to TSO mode. If it is desired to return back to URA after this command is issued, you must again use:

```
EXEC URA
```

Also, all the parameters previously assigned by the SET command are reset to their default values every time the EXEC URA statement is given.

The STOP command is used instead of the interrupt in order to delete a large number of working files created by URA. These datasets are used in processing URA Reports and should be deleted as a space saving policy.

Options and Alternatives

None

7.0 Modifier_Commands

All the commands in this section modify the URA database in some manner and generate an output to present the status of the modification. Each Modifier Command will be described in the following format:

Command name - (command abbreviation)

Modification made - what change is made in the database by this command.

Output description - presents such information as:
- Name of the output.

- Diagnostics given by the output.

Implementation - how to implement the modification.

Options and Alternatives - what variations in modifying the database can be made by usage of the command parameters.

Usage With NAME-GEN - how to aid modifications to the database by using NAME-GEN in conjunction with it.

Common errors - some of the common logical and syntactical errors that occur when implementing this command.

Examples are provided for the commands listed below.

7.1 CHANGE-TYPE (CT)

Modification Made

Each name specified as input to this command has its corresponding name type changed if the new name type does not conflict with the context in which the name has previously been used. This modification is most often implemented to change an undefined name (***UNKNOWN OR AMBIGUOUS***) to a specific name type (such as GROUP or ELEMENT).

Output Description

The output generated by this command is the CHANGE-TYPE REPORT. For each name used as input to the CHANGE-TYPE command, the name is printed out on the report followed by some diagnostics. If the modification is successful, the report will specify the old name type associated with that name and the new name type assigned to it. If the modification is not successful, error diagnostics will be given about why the attempt failed.

Implementation

To change the name type of only one name, the following command format is issued:

```
CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT
```

Previously in the user requirement, GROSS-PAY had been defined to be a GROUP. As it was more appropriate being an ELEMENT, the CHANGE-TYPE command made it easy to facilitate this change. The resulting CHANGE-TYPE REPORT is shown in Example 7.1a.

Options and Alternatives

It may be desirable to change several names at once via the FILE parameter. If a number of names are to have their types changed, this can be done as shown in Example 7.1b. Each of the three names put into CHANGE.DATA were previously undefined (not assigned a name type). The format for CHANGE.DATA was as below:

```
DATE GROUP

EMPLOYEE-DATA GROUP

VALID-T-CARD ELEMENT
```

The format is not important so long as the name and its new name type appear within the first eighty characters of the line and at least one blank appears between them.

It is also possible to change a list of names to the same type by use of the TYPE parameter. In this case the names to be changed to GROUP types are listed in a dataset. The change type command is then given as below:

```
CT FILE=CHNG.DATA      TYPE=GROUP
```

which will cause all of the names in the file to be changed to GROUP.

If the two formats are mixed, the TYPE parameter will have precedence that is if the dataset CHANGE.DATA shown above is used with the type command the assigned types in CHANGE.DATA will be ignored. For example, the report in Example 7.1c gives the results of the command:

```
CT FILE=CHANGE.DATA     TYPE=ELEMENT
```

Usage With NAME-GEN

It can sometimes be a great advantage to use NAME-GEN output as input to CHANGE-TYPE. Say, for example, that all the undefined names in your user requirement are really GROUP names. This change can be made in the following manner:

```
NAME-GEN UNDEFINED

CHANGE-TYPE FILE     TYPE=GROUP
```

The NAME-GEN command retrieves all the undefined names (specified by the parameter) and puts them into the file URANAMES. Then, the CHANGE-TYPE command takes its input from the file specified in the FILE parameter. Since no file is specified, it takes its input from URANAMES. The TYPE=GROUP parameter specifies that all names in the input file for this command be changed to GROUP names. Example 7.1d shows that the operation was successful.

Common Errors

Probably the most common error is attempting to assign a name type to a name that is used in another context. For example, if we previously stated that E1 was CONTAINED in SET S1, this would imply that E1 was an ENTITY. Now, if we attempted to change E1's name type to GROUP, an error message would be given notifying us that E1 has been used in a different context. It is illegal in URL to say a GROUP is contained in a SET. (See ESD TR# XXXXXXXXXX), "URL Users Manual" for the complete set of rules for using URL).

```
URA VERSION 740710                                        JUL 29, 1974  22:46.58
                              ADS-EXAMPLE
                          CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

   NAME=GROSS-PAY TYPE=ELEMENT

 1*  GROSS-PAY
   OLD TYPE - GROUP
   NEW TYPE - ELEMENT
```

Example 7.1a

```
      URA VERSION 740710                                               JUL 28, 1974  21:34.54
                                            ADS-EXAMPLE

                                         CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

   FILE

  1*  DATE
   OLD TYPE - ** UNKNOWN/AMBIG. **
   NEW TYPE - GROUP


  2*  EMPLOYEE-DATA
   OLD TYPE - ** UNKNOWN/AMBIG. **
   NEW TYPE - GROUP


  3*  VALID-T-CARD
   OLD TYPE - ** UNKNOWN/AMBIG. **
   NEW TYPE - ELEMENT
```

Example 7.1b

```
     URA VEkSION 740710                                  AUG 11, 1974   17:30.31

                              CHANGE-TYPE REPORT

PARAMETEkS FOk: CHANGE-TYPE

  FILE TYPE=ELEMENT

 1*  DATE
  OLD TYPE - GROUP
  NEW TYPE - ELEMENT


 2*  EMPLOYEE-DATA
  OLD TYPE - GROUP
  NEw TYPE - ELEMENT


 3*  VALID-T-CARD
  OLD TYPE - ELEMENT
  NEW TYPE - ELEMENT
```

Example 7.1c

```
URA VERSION 740328                                        JUL  7, 1974  18:19.48
                              ADS-EXAMPLE

                                  CHANGE-TYPE REPORT


PARAMETERS FOR: CHANGE-TYPE

   FILE TYPE=GROUP

 1*   DATE
   OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
   NEW TYPE - GROUP


 2*   EMPLOYEE-DATA
   OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
   NEW TYPE - GROUP


 3*   NEW-EMP-VALIDATION
   OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
   NEW TYPE - GROUP


 4*   VALID-T-CARD
   OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
   NEW TYPE - GROUP


 5*   VALID-TERM-INFO
   OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
   NEW TYPE - GROUP
```

Example 7.1d

7.2 DELETE (DEL)

Modification Made

For each name specified as input to the DELETE command, all its relationships with other names in the database are severed (i.e., USES, SUBPARTS, etc.), its comment entries (such as DESCRIPTION or PROCEDURE) are deleted and finally the name is deleted from the database.

Output Description

The DELETION REPORT is produced each time this command is initiated. It presents the status of the change for each name used as input. If the change is successful, the report will specify that the name is deleted; if not, it will generate error diagnostics.

Implementation

To delete one name from the database, the following command was given:

```
DELETE NAME=FIELD-CHECK-NEW
```

This name was deleted because the description of this PROCESS is already incorporated within another PROCESS and no longer necessary. The DELETION REPORT for this action is shown in Example 7.2a.

As with most other commands, it is possible to delete several names at one time by putting the list of names into a file.

```
EDIT DEL.DATA NEW NONUM

INPUT

stub

check
```

```
fixed-employee-data

(blank line RETURN)

EDIT

se

EXEC CLI

DELETE FILE=DEL.DATA
```

All these names will be deleted from the database as shown by Example 7.2b.

Usage With NAME-GEN

Though it is possible to use NAME-GEN and DELETE together, there is no general case where this would commonly be done.

Common Errors

Delete should not be used to delete the entire contents of a database. It is much more economical to simply delete the database and create a new one.

Another common error that can occur when doing minor editing of the database is to delete the name before saving the information connected to that name. What should be done first is a FORMATTED-PROBLEM-STATEMENT:

```
FPS NAME=name      PUNCH=PUNCH

DELETE NAME=name
```

At this point, the old information in PUNCH can be edited to suit the user and then re-entered via the INPUT-URL command:

```
INPUT-URL UPDATE INPUT=PUNCH
```

Also, if a dataset is used (INPUT=) as input to the command, all names to be deleted must begin in the first column of the file line. Any preceding blanks will be interpreted as being part of the name.

```
      URA VERSION 740710                                    AUG 11, 1974   17:30.31

                           D E L E T I O N   R E P O R T

PARAMETERS FOR: DELETE

   NAME=FIELD-CHECK-NEW

DELETED - FIELD-CHECK-NEW
```

Example 7.2a

```
      URA VERSION 740710                                              JUL 11, 1974   21:38.52
                                      ADS-EXAMPLE
                              D E L E T I O N   R E P O R T

PARAMETERS FOR: DELETE

   FILE

DELETED - STUB
DELETED - CHECK
DELETED - FIXED-EMPLOYEE-DATA
```

Example 7.2b

7.3 DELETE-COMMENT-ENTRY (DCOM)

Modification Made

The DELETE-COMMENT-ENTRY takes those names specified as input and deletes, for each input name, the comment entries associated with those comment entry types designated as command parameters.* If no comment entry types are specified by the parameters, no comment entries will be deleted.

Output Description

The output report generated by this command is called DELETED COMMENT ENTRIES. This report gives a status listing of every name used as input to the command. This status listing either prints out each comment entry type and corresponding comment entry that was deleted (for each input name) or some error diagnostic to why there was no deletion.

Implementation

To delete the PROCEDURE comment entries associated with one name, I gave:

```
DCOM N=NEW-INFO-VALIDATION PRCD
```

This was done because the current PROCEDURE comment entry was no longer valid and there was no replacement. If the comment entry could be correct if edited or a replacement was available, then it would be more appropriate to use the REPLACE-COMMENT-ENTRY command (see Section 7.7). The output report generated by this command is shown in Example 7.3a.

Options and Alternatives

1. Multiple comment entries could be deleted for a name:

   ```
   DCOM N=NEW-INFO-VALIDATION PRCD DESC
   ```

   In this case, the PROCEDURE and DESCRIPTION comment entries would be deleted for NEW-INFO-VALIDATION.

* An example of a comment entry type is a URL "DESCRIPTION" or "PROCEDURE" statement. The comment entry associated with this comment entry type would be the text specified by the user for the particular statement.

2. Several names could have a comment entry deleted such as:

```
EDIT DCOM.DATA NEW NONUM
INPUT
NEW-EMPLOYEE-PRINTING
NEW-INFO-VALIDATION
(blank line RETURN)
EXEC CLI
     DCOM FILE=DCOM.DATA DESC
```

This would mean that all the names in the file, DCOM.DATA, would have their DESCRIPTION comment entries deleted.

3. Several names could have multiple comment entries deleted:

```
EDIT DCOM.DATA NEW
INPUT
NEW-EMPLOYEE-PRINTING
NEW-INFO-VALIDATION
(blank line RETURN)
EXEC CLI
     DCOM FILE=DCOM.DATA DESC PRCD
```

The output report generated from this command is shown in Example 7.3b. It is not necessary that all the names in the input-file have both DESCRIPTION and PROCEDURE comment entries. The command will delete only the specified comment entries that exist for each name. A message is given, however, that the comment entry did not exist for the name.

Common Errors

It is important to delete only those specific comment entries that are intended to be deleted.

```
     URA VERSION 740710                                                     JUL 28, 1974   21:34.54
                                        ADS-EXAMPLE

                             D F L E T E D   C O M M E N T   E N T R I E S

PARAMETERS FOR: DCOM

  NAME NODESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
  NOTRUE-WHILE NOFALSE-WHILE PRINT NOFILE


1*    NEW-INFO-VALIDATION
              PROCEDURE;
         1                 1-) READ A UNIT OF NEW EMPLOYEE INFORMATION
         2                 2-) CHECK THE RANGES OF THE FIELDS
         3                 3-) IF: FIELDS CORRECT
         4                       THEN: ADD TO THE DATA BASE
         5                       ELSE: REJECT ENTIRE UNIT OF INFORMATION                       ;
```

Example 7.3a

```
     URA VERSION 740328                                                    JUN  8, 1974  14:04.08

                              D E L E T E D   C O M M E N T   E N T R I E S

PARAMETERS FOR: DCOM

   DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT FILE


1*    NEW-EMPLOYEE-PRINTING
              DESCRIPTION;
         1                THIS PROCESS PRODUCES THE NEW HIRE SECTION OF THE H-T REPORT.;


2*    NEW-EMPLOYEE-PRINTING
              PROCEDURE;
         1                1-) ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
         2                2-) SAVE THE INFORMATION
         3                3-) PRINT THE NEW HIRE SECTION OF THE HT REPORT.             ;


3*    NEW-INFO-VALIDATION
              DESCRIPTION;
         1                THIS PROCESS ACCEPTS CORRECT INPUT INFORMATION AND
         2                REJECTS THE INPUT OTHERWISE.                                 ;


4*    NEW-INFO-VALIDATION
              PROCEDURE;
         1                1-) READ A UNIT OF NEW EMPLOYEE INFORMATION
         2                2-) CHECK THE RANGES OF THE FIELDS
         3                3-) IF: FIELDS CORRECT
         4                        THEN: ADD TO THE DATA BASE
         5                        ELSE: REJECT ENTIRE UNIT OF INFORMATION              ;
```

Example 7.3b

7.4 DELETE-URL (DURL)

Modification Made

This command takes as input, any URL statements in the format specified in the URL Users Manual (ESD TR#XXXXXXXXXX). For each section header statement (i.e., PROCESS, DEFINE, etc), all the URL statements following this section header (up to the next section header statement) will be deleted from the URA database for those names specified in the header statement. This command only deletes relationships between names and does not delete any defined names from the database. It also does not delete any comment entries (this is handled by the DCOM command). If some of the information presented by the URL statements is contradictory, an error message will be given for that statement. Error diagnostics are also given when syntactical errors occur. URA attempts to continue the procedure until too many errors are encountered. (See Section 9 for the limit of errors allowed).

Output Description

The outputs by this command are DELETED URL and the URA CROSS REFERENCE. The DELETED URL output presents all the URL statements used as input into the command exactly as they were inputted. Error diagnostics are also produced at appropriate points in the output. These aid in correcting any errors that occur.

The URA CROSS REFERENCE is an index to the DELETED URL. It is only produced at request (i.e., by specifying XREF as a command parameter) and produces an alphabetical listing of all names used in the DELETED URL output, their corresponding name types, and the line numbers in DELETED URL where each name can be referenced.

Implementation

The most common method of deleting URL statements is by first writing all statements to be deleted into a dataset and then using this file as input to the command via the INPUT parameter.

```
EDIT DEL.DATA NONUM NEW

INPUT

group check;

contained pay-statement;

relation comp-pay-info;

syn comp-pay-info;

element payrate;

     attr type numberic-information;

     values 3 thru 100;

eof

(blank line RETURN)

EDIT

se

EXEC CLI

DELETE-URL I=DEL.DATA
```

The I=DEL.DATA specifies that the input to this command should be read from the dataset DEL.DATA. Example 7.4a presents the DELETED URL output for this procedure.

<u>Options and Alternatives</u>

1. In many cases when the amount of input is relatively large (say, a few hundred lines) there may be a need for the URA CROSS REFERENCE. By

specifying the XREF parameter with the command, it will be generated.

2. If no input dataset is given, URA will wait for URL statements from the terminal. In this manner, most errors occurring in the input procedure may be corrected interactively. Each line entered by the user is echoed back along with any diagnostics for errors which have been encountered within the previously entered line. This method of inputting information is useful from the error correcting standpoint, but is hardly feasible if there are hundreds or thousands of input lines.

Common Errors

The most common errors are typing errors. A typing mistake can cause many different types of syntactical and semantic errors.

Only the first 72 columns of each line in the input file are read so all URL statements should fit in this region. Anything over 72 will be truncated and an error message will be generated in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors.

DELETE-URL will not delete comment entry statements from the database so these statements are ignored if encountered in the input file. Names cannot be deleted from the database either. Only SYNONYMS for names can be deleted.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. (See Example 6.4a to see how EOF is used correctly). No URL statements are read after EOF.

```
     URA VERSION 740710                                                  AUG 12, 1974  09:10.20
                                          ADS-EXAMPLE

                                          DELFTED URL

PARAMETERS FCR: DPSL

   SOURCF NOXREF

  LINE   S T M T                                                                        ID FIELD

     1 >GRCUP CHECK;                                                                   < >      <
     2 > CONTAINED PAY-STATEMENT;                                                      < >      <
     3 >RELATICN COMP-PAY-INFO;                                                        < >      <
     4 > SYN COMP-PAY-INFO;                                                            < >      <
     5 >ELEMENT PAYRATE;                                                               < >      <
     6 > ATTR TYPE NUMERIC-INFCRMATION;                                                < >      <
     7 > VALUES 3 THRU 100;                                                            < >      <
     8 >EOF                                                                            < >      <
```

Example 7.4a

7.5 INPUT-URL (IP)

Modification Made

This command takes as input, any URL statements in the format specified in the URL Users Manual (ESD TR #XXXXXXXXXX). For each section header statement (i.e., PROCESS, DEFINE, etc.) the user defined name specified by that section header will be added to the list of names in the database (if not in already). All the URL statements following this section header, up to the next section header statement, specify connections to be made with other names in the database. If some of the information presented by the URL statements is contradictory, an error message will be given for that statement. Error diagnostics are also given when syntactical errors occur. URA attempts to continue the input procedure until too many errors are encountered. (See Section 9 for the limit of errors allowed).

Output Description

The outputs generated by this command are the URA AS-IS SOURCE LISTING and the URA CROSS REFERENCE.

The URA AS-IS SOURCE LISTING presents all the URL statements used as input into the command exactly as they were inputted. Error diagnostics are also produced at appropriate points in the LISTING. These aid in correcting any errors that occur.

The URA CROSS REFERENCE is an index into the URA AS-IS SOURCE LISTING. It is only produced at request (i.e., by specifying XREF as a command parameter) and produces an alphabetical listing of all names used in the AS-IS LISTING, their corresponding name types (explicitly or implicitly defined) and the line numbers in the AS-IS LISTING each name can be referenced.

Implementation

The most common method of inputting URL statements is by first creating a data set, writing all URL statements into this dataset and then using this dataset as an input via the INPUT parameter. For example,

```
EDIT INP.DATA NONUM NEW

INPUT

process field-check-new;

    part of:  new-employee-updating;

    utilized by:  new-info-validation;

eof

(blank line RETURN)

EDIT

se

EXEC CLI

INPUT-URL     I=INP.DATA UPDATE
```

The UPDATE parameter specifies that the URA database is to be modified by the input. If this parameter is not given the information contained in INP.DATA will be semantically and syntactically checked against the URA database but will not change the database. Example 7.5a presents the AS-IS LISTING for this procedure.

Options and Alternatives

1. In many cases, when the amount of input is relatively large (say, a few hundred lines) there may be a need for the CROSS REFERENCE. By simply specifying, XREF, as a parameter it will be generated. Example 7.5b presents an AS-IS LISTING and CROSS REFERENCE for a small problem statement.

2. In most cases, it is advantageous to first do a syntax and semantic check of the input data before you attempt to put it in the database. By not specifying UPDATE, these checks will be made without actually putting the information into the database. The command:

```
INPUT-URL      I=INP.DATA
```

will generate as AS-IS LISTING with error diagnostics for the information in the dataset. Since most problem statements have one or two typing errors anyway, this proves to be an inexpensive way to catch errors early. After the source of the errors has been determined and corrected, the command can be called again using UPDATE as a parameter.

3. If no input dataset is given, URA will wait for URL statements from the terminal. In this manner, most errors occurring in the input procedure may be corrected interactively. Each line entered by the user is echoed back along with any diagnostics for errors which have been encountered within the previously entered line. This method of inputting information is useful from the error correcting standpoint, but is hardly feasible if there are hundreds or thousands of input lines.

Common Errors

The most common errors are typing errors. A typing mistake can cause so many different types of syntactical and semantic errors that it will be handled in a later section (Section 10).

Inputting the new information into the wrong database also often happens. Be sure to: SET DB=dbname.database when entering URA mode. If this is not done, the data base will be set to: DB=URADB.DATABASE which may or may not be the database you want to access.

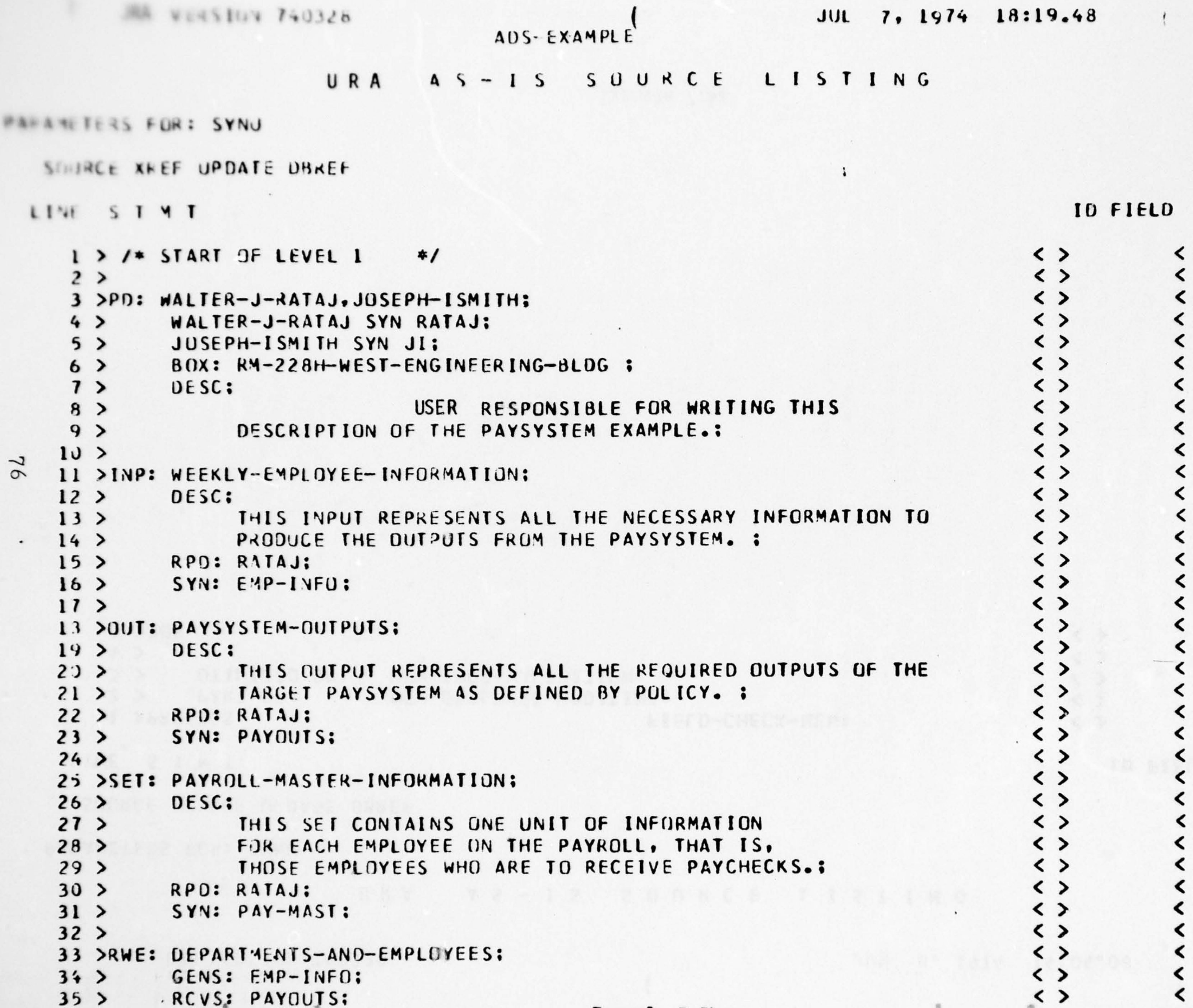
The INPUT-URL command only reads the first 72 columns of each line in the input file so all URL statements must fit in this region. Anything over 72 will be truncated and an error message will be given in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. (See Examples 7.5a and 7.5b to see how EOF is used correctly). No URL statements are read after EOF.

```
     URA VERSION 740328                                               JUN  8, 1974  14:04.08

                                URA   AS-IS   SOURCE   LISTING

PARAMETERS FOR: SYNU

  SOURCE NOXREF UPDATE DBREF

  LINE  STMT                                                                             ID FIELD

     1 >PROCESS                          FIELD-CHECK-NEW;                          < >          <
     2 >     PART OF:     NEW-EMPLOYEE-UPDATING;                                   < >          <
     3 >     UTILIZED BY: NEW-INFO-VALIDATION;                                     < >          <
     4 >                                                                           < >          <
     5 >EOF                                                                        < >          <
```

Example 7.5a



```
URA VERSION 740328                                          JUL  7, 1974  18:19.48
                              ADS-EXAMPLE

                    U R A   A S - I S   S O U R C E   L I S T I N G

PARAMETERS FOR: SYNU

  SOURCE XREF UPDATE DBREF

 LINE  S T M T                                                              ID FIELD

    1 > /* START OF LEVEL 1     */                                     < >          <
    2 >                                                                < >          <
    3 >PD: WALTER-J-RATAJ,JOSEPH-ISMITH;                                < >          <
    4 >     WALTER-J-RATAJ SYN RATAJ:                                  < >          <
    5 >     JOSEPH-ISMITH SYN JI;                                      < >          <
    6 >     BOX: RM-228H-WEST-ENGINEERING-BLDG ;                       < >          <
    7 >     DESC;                                                      < >          <
    8 >                        USER  RESPONSIBLE FOR WRITING THIS      < >          <
    9 >          DESCRIPTION OF THE PAYSYSTEM EXAMPLE.:                < >          <
   10 >                                                                < >          <
   11 >INP: WEEKLY-EMPLOYEE-INFORMATION;                               < >          <
   12 >     DESC;                                                      < >          <
   13 >          THIS INPUT REPRESENTS ALL THE NECESSARY INFORMATION TO< >          <
   14 >          PRODUCE THE OUTPUTS FROM THE PAYSYSTEM. ;             < >          <
   15 >     RPD: RATAJ;                                                < >          <
   16 >     SYN: EMP-INFO;                                             < >          <
   17 >                                                                < >          <
   13 >OUT: PAYSYSTEM-OUTPUTS;                                         < >          <
   19 >     DESC;                                                      < >          <
   20 >          THIS OUTPUT REPRESENTS ALL THE REQUIRED OUTPUTS OF THE< >          <
   21 >          TARGET PAYSYSTEM AS DEFINED BY POLICY. ;              < >          <
   22 >     RPD: RATAJ;                                                < >          <
   23 >     SYN: PAYOUTS;                                              < >          <
   24 >                                                                < >          <
   25 >SET: PAYROLL-MASTER-INFORMATION;                                < >          <
   26 >     DESC;                                                      < >          <
   27 >          THIS SET CONTAINS ONE UNIT OF INFORMATION             < >          <
   28 >          FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,            < >          <
   29 >          THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;        < >          <
   30 >     RPD: RATAJ;                                                < >          <
   31 >     SYN: PAY-MAST;                                             < >          <
   32 >                                                                < >          <
   33 >RWE: DEPARTMENTS-AND-EMPLOYEES;                                 < >          <
   34 >     GENS: EMP-INFO;                                            < >          <
   35 >     RCVS; PAYOUTS;                                             < >          <
```

Example 7.5b

```
   URA VERSION 740328                                              JUL  7, 1974  18:19.48
                                  ADS-EXAMPLE

                     U R A    A S - I S   S O U R C E   L I S T I N G

LINE  S T M T                                                                      ID FIELD

 36 >       SYN: DEPT-EMP:                                                     < >        <
 37 >       DESC:                                                              < >        <
 38 >            THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND     < >        <
 39 >        SUPPLY ALL THE INPUTS. ;                                          < >        <
 40 >                                                                          < >        <
 41 >PRC: PAYROLL-PROCESSING;                                                 < >        <
 42 >       UPDS PAY-MAST;                                                     < >        <
 43 >       RCVS: EMP-INFO;                                                    < >        <
 44 >       GENS: PAYOUTS;                                                     < >        <
 45 >       DESC:                                                              < >        <
 46 >            THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS             < >        <
 47 >            IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES               < >        <
 48 >            ALL INPUTS AND PRODUCES ALL OUTPUTS.;                         < >        <
 49 >       KEY: LEVEL-1, TARGET-SYSTEM, HIGHEST-LEVEL-PROCESS;                < >        <
 50 >       RPD: RATAJ;                                                        < >        <
 51 >       SYN: PAYPROC:                                                      < >        <
 52 >       SOURCE: COMPANY-PROCEDURES-MANUAL;                                 < >        <
 53 >                                                                          < >        <
 54 > /* END OF LEVEL 1                                           */           < >        <
 55 >           EOF                                                            < >        <
```

Example 7.5b (Cont'd)

```
     URA VERSION 740328                                         JUL  7, 1974  13:19.48
                                  ADS-EXAMPLE

                             U R A   C R O S S   R E F E R E N C E

SEQ N A M E                       T Y P E

  1 COMPANY-PROCEDURES-MANUAL     SOURCE
                              52

  2 DEPARTMENTS-AND-EMPLOYEES     REAL-WORLD-ENTITY
                              33

  3 DEPT-EMP                      SYNONYM FOR DEPARTMENTS-AND-EMPLOYEES
                              36

  4 EMP-INFO                      SYNONYM FOR WEEKLY-EMPLOYEE-INFORMATION
                              16   34   43

  5 HIGHEST-LEVEL-PROCESS         KEYWORD
                              49

  6 JI                            SYNONYM FOR JOSEPH-ISMITH
                               5

  7 JOSEPH-ISMITH
                               3    5

  8 LEVEL-1                       KEYWORD
                              49

  9 PAY-MAST                      SYNONYM FOR PAYROLL-MASTER-INFORMATION
                              31   42

 10 PAYOUTS                       SYNONYM FOR PAYSYSTEM-OUTPUTS
                              23   35   44

 11 PAYPROC                       SYNONYM FOR PAYROLL-PROCESSING
                              51

 12 PAYROLL-MASTER-INFORMATION    SET
                              25

 13 PAYROLL-PROCESSING            PROCESS
                              41
```

Example 7.5b (Cont'd)

```
       URA VERSION 740328                                        JUL  7, 1974  18:19.48
                                   ADS-EXAMPLE

                            U R A  C R O S S   R E F E R E N C E

SEQ N A M F                        T Y P E

 14 PAYSYSTEM-OUTPUTS              OUTPUT
                              18

 15 RATAJ                          SYNONYM FOR WALTER-J-RATAJ
                               4    15    22    30    50

 16 RM-228H-WEST-ENGINEERING-BLDG  MAILBOX
                               6

 17 TARGET-SYSTEM                  KEYWORD
                              49

 18 WALTER-J-RATAJ                 USER
                               3    4

 19 WEEKLY-EMPLOYEE-INFORMATION    INPUT
                              11
```

Example 7.5b (Cont'd)

7.6 RENAME (REN)

Modification Made

The RENAME command takes an old name (of some object in the problem defineds data base) and a new name as input. If the new name is not a URL reserved word or a name already in the data base, the command will replace the old name by the new name.

Output Description

The output generated from this command is called the RENAME REPORT. If the change is successful, the old name and new name are printed under their respective report headings. If an error is encountered, error diagnostics will be given for that particular old name-new name pair.

Implementation

To change one name in the data base, the following command was given:

```
     RENAME  OLD=EMPLOYEE-CODE  NEW=EMPLOYEE-NUMBER
```

Upon first defining the parget system, EMPLOYEE-CODE was used to represent a certain piece of data. Later it was found that this data was actually called EMPLOYEE-NUMBER and made the change to be consistent with organizational terminology. See Example 7.6a for the report generated by this command. This command is also beneficial for changing mispelled names in the data base. Through typing errors, EMPLOYEE-NUMBER may have gone in as EMPLYEE-NUBER. This mistake can be corrected by:

```
     RENAME  OLD=EMPLYEE NUBER  NEW=EMPLOYEE-NUMBER
```

Options and Alternatives

As with most of the modifier commands, the problem definer has the option of changing several names at one time. The old name-new name pairs must first be put in a file to be used as input to the command. The procedure could be done like this:

```
EDIT REN.DATA NONUM NEW
INPUT
JOSEPH-SMITH   HENRY-MILLER
VARYING-EMPLOYEE-DATA CHANGING-EMPLOYEE-DATA
LEVEL-1   L1
ERROR-LISTING  ERROR LIST
(blank line RETURN)
EDIT
SE
EXEC CLI
     RENAME INPUT=REN.DATA
```

The output of this procedure is shown in Example 7.6b.

The old name is specified first on each line, followed by its new name. Free format allows the two names to be anywhere in the first eighty columns of the file line as long as the old name comes first and the two names are separated by at least one blank.

Usage With NAME-GEN

The two commands could not be used together directly. Additional preparation would be needed, discussion of which is beyond the schope of this paper.

Common Errors

The most common error in using RENAME is specifying a name already in the data base or a URL reserved word as the new name. URA will not make the change if this is the case. The command would have to be reissued with another new name to take the place of the illegal one.

```
URA VERSION 740328                                     JUN  8, 1974  14:04.08

                                   RENAME REPORT

PARAMETERS FOR: REN

   OLD=EMPLOYEE-CODE NEW=EMPLOYEE-NUMBER

SEQ OLD NAME                                 NEW NAME
  1 EMPLOYEE-CODE                            EMPLOYEE-NUMBER
```

Example 7.6a

```
URA VERSION 740328                                        JUL  6, 1974  16:01.44
                          ADS-EXAMPLE

                             RENAME REPORT

PARAMETERS FOR: REN

  INPUT

SEQ OLD NAME                      NEW NAME
  1 JOSEPH-ISMITH                 HENRY-MILLER
  2 VARYING-EMPLOYEE-DATA         CHANGING-EMPLOYEE-DATA
  3 LEVEL-1                       L1
  4 ERROR-LISTING                 ERROR-LIST
```

Example 7.6b

7.7 REPLACE-COMMENT-ENTRY (RCOM)

Modification Made

This command takes as input, names which exist in the data base, each followed by a URL comment entry statement. If the comment is a DESCRIPTION comment entry, for example, then the command will replace the old DESCRIPTION comment entry by the one used as input. If there is no old DESCRIPTION to replace, the new one will still be given to the name and a warning will be generated on the output report.

Output Description

The output generated by this report is called REPLACED COMMENT ENTRIES. For each name-comment entry statement pair the report prints out the name, the comment entry in the data base which has been deleted and the comment entry which replaces it.

Implementation

To change a couple of comment entries associated with a particular name, the information must first be put into a file:

```
EDIT RCM.DATA NONUM NEW
INPUT
NEW-EMPLOYEE-PRINTING
DESC;
This PROCESS PRODUCES THE NEW HIRE SECTION OF
THE H-T REPORT.
NEW-EMPLOYEE-PRINTING
PRCD:
1)  ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
2)  SAVE THE INFORMATION
3)  PRINT THE NEW HIRE SECTION OF THE H-T REPORT.
(blank line RETURN)
SE
EXEC CLI
     RCOM I=RCM.DATA
```

Doing this results in giving NEW-EMPLOYEE-PRINTING new DESCRIPTION and PROCEDURE comment entries. This was done because the previous DESCRIPTION was inaccurate and the PROCEDURE was not complete. The output generated by this procedure is shown by Example 7.7a.

Options and Alternatives

It is often the case that only some minor editing of a comment entry need be done to make it correct. This editing can be done easily by the following sequence of commands:

```
PCOM N=NEW-EMPLOYEE-PRINTING  DESC P=PUNCH
!
EDIT PUNCH
CHANGE 3 /ADDS INFORMATION TO/PRODUCES THE NEW HIRE SECTION OF/
SE
EXEC CLI
RCOM I=PUNCH
```

This method of replacing comment entries saves a lot of typing effort, especially when the comment entries are lengthy.

Common Errors

The major problem in using this command is specifying the file format correctly. It follows the pattern:

```
     name
     comment entry type;
       :
     comment entry
       :                ;
     name
     comment entry type;
       :
     comment entry
       :                ;
     etc.
     $ENDFILE
```

The correct format is also shown in the example on the previous page.

Though the command allows free formatting of the file, the order: name, comment-entry type, and comment entry must be maintained.

```
     URA VERSION 740710                                             JUL 29, 1974  22:46.58
                                          ADS-E    LF
                            REPLACED   COMMENT   ENTRIES

  PARAMETERS FOR: RCOM

     PRINT


 ** DELETED COMMENT ENTRY **
   1*    NEW-EMPLOYEE-PRINTING
                  DESCRIPTION ;
          1             THIS PROCESS ADDS INFORMATION TO THE  H-T REPORT              ;

 ** INSERTED COMMENT ENTRY **
   1*    NEW-EMPLOYEE-PRINTING
                  DESCRIPTION ;
          1             THIS PROCESS PRODUCES THE NEW HIRE SECTION OF THE H-T REPORT;


 ** DELETED COMMENT ENTRY **
   2*    NEW-EMPLOYEE-PRINTING
                  PROCEDURE ;
          1             1-) ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
          2             3-) PRINT THE NEW HIRE SECTION OF THE HT REPORT.              ;

 ** INSERTED COMMENT ENTRY **
   2*    NEW-EMPLOYEE-PRINTING
                  PROCEDURE ;
          1             1-) ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
          2             2-) SAVE THE INFORMATION
          3             3-) PRINT THE NEW HIRE SECTION OF THE HT REPORT.;
```

Example 7.7a

8. Report Commands

All the commands in this section generate reports and will be described in the following format:

Command name (command abbreviation)

Report Description* - presents such information as:

- Name of the report(s) generated by this command.
- The different URL name types this report can be produced for.
- What information is generated for each name type and name in the URA data base.

Implementation - how to obtain the basic report.

Options and Alternatives - what variations in format and content of the output report can be obtained through usage of the command parameters.

Usage with NAME-GEN - how to aid report production by using the NAME-GEN command in conjunction with it.

Common Errors - some of the common logical and syntactical errors that occur when implementing this command.

Examples

*For a more detailed description of the reports generated by these commands, see Part III ,"URA OUTPUTS" .

8.1 CONSISTS-COMPARISON (CNC)

Report Description

The CONSISTS COMPARISON REPORT is generated by this command. This report retrieves structure information about the data objects (SETS, INPUTS, OUTPUTS, ENTITIES and/or GROUPS) used as input to the command. For each input name, the command derives its lowest level constituents (other data names) via the CONSISTS statement relationships specified in the data base. A matrix is used to present this information. The names used as input to the command are represented by the rows of the matrix and the low level constituents, by the columns. A count of the constituents for each input name and the number of constituents in common between any two input names is presented in a second matrix. In this matrix, each input name is represented by a column and row.

Implementation

The most common form of implementing this command is:

```
NG ENTITY GROUP
CNC
```

This produces the report shown by Example 8.1a. NAME-GEN is first used to retrieve all GROUP and ENTITY names in the data base. "CNC" is then specified and so the names retrieved by NAME-GEN are used as input to the command.

Options and Alternatives

The only parameter available for this command is the "FILE" parameter which can be used in the following manner:

```
EDIT CNC.DATA NONUM NEW
F-DATA
V-DATA
(blank line RETURN)
SE
EXEC CLI
     CNC F=CNC.DATA
```

which will generate the report for only these two names. Since the report is rather large, anyway, and relatively expensive to generate, it is best to have the report generated for several names (using NAME-GEN) at one time.

Usage With NAME-GEN

(See the example specified above.)

Common Errors

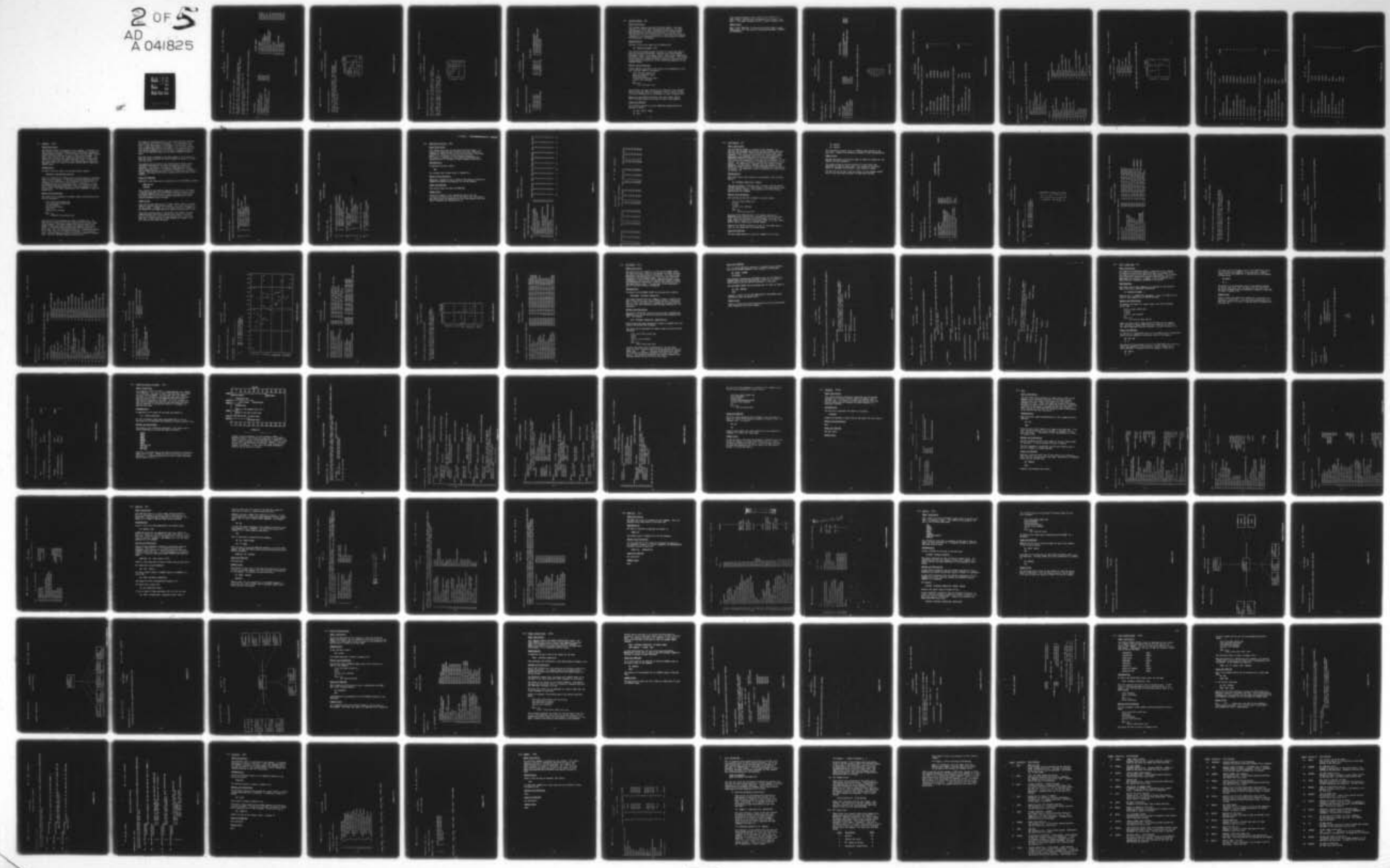
It must be kept in mind that only data object names (SETS, INPUTS, OUTPUTS, ENTITIES and GROUPS) can be used as input to the command.

```
      URA VERSION 74071J                      (                      AUG 12, 1974  20:54.50        (
                                         ADS-EXAMPLE
                                   CONSISTS COMPARISON REPORT
PARAMETERS FCR: CNC
   FILE

   URA 273:CNCBLD: NAME DOESNT CONSIST OF ANYTHING -  BAD-INPUT-DATA

   URA 273:CNCBLD: NAME DCESNT CONSIST OF ANYTHING -  CHECK

   URA 273:CNCBLD: NAME DOESNT CONSIST OF ANYTHING -  EXPLICIT-PERSONAL-DATA

   URA 273:CNCBLD: NAME DOESNT CONSIST OF ANYTHING -  RELATED-PERSONAL-DATA
```

Example 8.1a

AD-A041 825 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370--ETC(U)
APR 75 C R MOORE, H J EIDEN
UNCLASSIFIED ESD-TR-75-88-VOL-3 NL

2 OF 5
AD
A 041825



```
URA VERSION 740710                                         AUG 12, 1974  20:54.50
                                  ADS-EXAMPLE

                            CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

THE ROWS ARE THE GIVEN INPUT NAMES.

THE COLUMNS ARE THE LOWEST LEVEL OBJECTS WHICH ARE
CONTAINED IN THE ROWS, WITH INTERMEDIATE GROUPS IGNORED.

IF ANY COLUMNS ARE GROUP NAMES, THEN THE
DEFINITION IS INCOMPLETE.

IF ANY COLUMNS ARE AMBIGUOUS NAMES, THEY ARE POSSIBLE ELEMENTS.


  ROW NAMES                                     COLUMN NAMES

1 EMPLOYEE-NAME                  GROUP           1 FIRST-NAME                    ELEMENT
2 FIXED-EMPLOYEE-DATA            ENTITY          2 LAST-NAME                     ELEMENT
3 NEW-EMPLOYEE-PART              GROUP           3 MIDDLE-INITIAL                ELEMENT
4 STUB                           GROUP           4 EXPLICIT-PERSONAL-[illegible] GROUP
5 TERMINATED-EMPLOYEE-PART       GROUP           5 EMPLOYEE-NUMBER               ELEMENT
6 VALID-NEW-INFO                 GROUP           6 RELATED-PERSONAL-DATA         GROUP
7 VARYING-EMPLOYEE-DATA          ENTITY          7 NUMBER-OF-DEPENDENTS          ELEMENT
                                                 8 PAYRATE                       ELEMENT
                                                 9 DEDUCTIONS                    ELEMENT
                                                10 GROSS-PAY                     ELEMENT
                                                11 NET-PAY                       ELEMENT
                                                12 TERMINATION-CODE              ELEMENT
                                                13 TAX-RATE                      ELEMENT
                                                14 YTD-DEDUCT                    ELEMENT
                                                15 YTD-GROSS                     ELEMENT
```

Example 8.1a (cont'd)

```
URA VEHSIUN 74071C                                              AUG 12, 1974  20:54.50
                                  ADS-EXAMPLE

                         CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

AN * IN (I,J) MEANS THAT COLUMN J IS CONTAINED
DIRECTLY UK INDIRECTLY IN ROW I.  THE COLUMNS
DO NOT CONSISTS OF ANYTHING FURTHER.  INTERMEDIATE
GRUUPS ARE IGNORED.

                                       1 11111
                              1234567890 12345
                             +----------+-----+
                           1 I***       I     I
                           2 I******    I     I
                           3 I*** * **  I     I
                           4 I    *  ***I*    I
                           5 I*** *     I *   I
                             +----------+-----+
                           6 I*** * **  I     I
                           7 I    * **  I  ***I
                             +----------+-----+
```

Example 8.1a (cont'd)

```
URA VERSION 740710                                             AUG 12, 1974  20:54.50
                                       ADS-EXAMPLE

                                CONSISTS COMPARISON REPORT

CONTENTS SIMILARITY MATRIX

THE NUMBER IN (I,I) IS THE NUMBER OF OBJECTS
AT THE LOWEST LEVEL CONTAINED IN ROW I FROM ABOVE.

THE NUMBER IN (I,J) (I NOT EQUAL J) IS
THE NUMBER OF OBJECTS AT THE LOWEST LEVEL IN
COMMON BETWEEN ROWS I AND J FROM ABOVE.


                 1 2 3 4 5  6 7
               +----------+----+
             1 I 3 3 3   3I 3  I
             2 I   6 4 1 4I 4 1I
             3 I     6 2 4I 6 3I
             4 I       5 1I 2 2I
             5 I         5I 4 1I
               +----------+----+
             6 I          I 6 3I
             7 I          I   6I
               +----------+----+
```

Example 8.1a (cont'd)

```
URA VERSION 740710                                                    AUG 12, 1974  20:54.50
                                      ADS-EXAMPLE

                               CONSISTS COMPARISON REPORT

CONTENTS SIMILARITY SUMMARY

ROW# NAME                                             ROW# NAME

   1 EMPLOYEE-NAME                  IS A SUBSET OF      2 FIXED-EMPLOYEE-DATA
   1 EMPLOYEE-NAME                  IS A SUBSET OF      3 NEW-EMPLOYEE-PART
   1 EMPLOYEE-NAME                  IS A SUBSET OF      5 TERMINATED-EMPLOYEE-PART
   1 EMPLOYEE-NAME                  IS A SUBSET OF      6 VALID-NEW-INFO
   3 NEW-EMPLOYEE-PART            IS EQUIVALENT TO      6 VALID-NEW-INFO
```

Example 8.1a (cont'd)

8.2 CONSISTS-MATRIX (CM)

Report Description

This command produces the CONSISTS MATRIX REPORT. The report consists of a matrix where the names represented by the rows are CONTAINED in the names represented by the columns through CONSISTS/CONTAINED relationships specified in the data base. The existence of such a relationship is specified by an asterisk. A summary is also produced which aids in describing the information portrayed in the matrix.

Implementations

The basic form of the report can be obtained by:

```
CM  N=EMPLOYEE-NUMBER  CNTD
```

This particular command format retrieves all these data objects which contain EMPLOYEE-NUMBER. Whenever using the CONTAINED parameter (CNTD), all the names used as input to the command must be ELEMENT, GROUP, INPUT, OUTPUT and/or ENTITY names. The information presented for EMPLOYEE-NUMBER is retrieved from the data base in the opposite direction as is the information presented in the CONTENTS REPORT.

Options and Alternatives

A more commonly used form of the report can be obtained by using the "FILE"and "CONSITS" parameters:

```
EDIT CM.DATA NONUM NEW
EXPLICIT-PESONAL-DATA
EMPLOYEE-NAME
TERMINATED-EMPLOYEE-PART
(blank line RETURN)
SE
EXEC CLI
      CM F=CM.DATA CSTS
```

This produces the same information as presented in the CONTENTS REPORT (using the LEVELS=-1 parameter), but in a matrix format. The report generated by this procedure is shown in Example 8.2a.

When using the CONSISTS parameter, only SET, INPUT, OUTPUT, ENTITY and GROUP names may be used as input to the command.

Usage With NAME-GEN

It is usually easiest to use the NAME-GEN command with CM to generate a report:

```
NG  SET  ENTITY  GROUP
CM  CSTS
```

This sequence produces a matrix with all SET, ENTITY and GROUP names in the data base representing the columns of the matrix. The report produced by this is shown in Example 8.2b.

Common Errors

What is most important is that only the proper types of data names (SETS, etc.) are used with the proper parameter (CONSISTS or CONTAINED).

```
   URA VERSION 740710                                                      AUG 12, 1974  20:54.50
                                        ADS-EXAMPLE

                                  CONSISTS MATRIX REPORT

PARAMETERS FOR: CM

   FILE CONSISTS


   URA 315:CONCOL: THE FOLLOWING DO NOT CONSIST OF ANYTHING:

        EXPLICIT-PERSONAL-DATA


      ROW NAMES                                                      COLUMN NAMES

   1 FIRST-NAME                    ELEMENT                         1 EXPLICIT-PERSONAL-DATA        GROUP
   2 LAST-NAME                     ELEMENT                         2 EMPLOYEE-NAME                 GROUP
   3 MIDDLE-INITIAL                ELEMENT                         3 TERMINATED-EMPLOYEE-PART      GROUP
   4 EMPLOYEE-NAME                 GROUP
   5 TERMINATION-CODE              ELEMENT
   6 EMPLOYEE-NUMBER               ELEMENT

                         THE ROWS ARE CONTAINED IN THE COLUMNS WITH *S




                                        123
                                       +---+
                                     1 I * I
                                     2 I * I
                                     3 I * I
                                     4 I  *I
                                     5 I  *I
                                       +---+
                                     6 I  *I
                                       +---+
```

Example 8.2a

```
URA VERSION 740710                                              AUG 12, 1974  20:54.50
                                     ADS-EXAMPLE
                                CONSISTS MATRIX REPORT
**THE NUMBER OF COLUMNS THAT CONTAIN THE ROWS**

     ROW                            TYPE                  COUNT

  1  FIRST-NAME                     ELEMENT                   1

  2  LAST-NAME                      ELEMENT                   1

  3  MIDDLE-INITIAL                 ELEMENT                   1

  4  EMPLOYEE-NAME                  GROUP                     1

  5  TERMINATION-CODE               ELEMENT                   1

  6  EMPLOYEE-NUMBER                ELEMENT                   1


**THE NUMBER OF ROWS CONTAINED IN THE COLUMNS**

     COLUMN                         TYPE                  COUNT

  2  EMPLOYEE-NAME                  GROUP                     3

  3  TERMINATED-EMPLOYEE-PART       GROUP                     3

  1  EXPLICIT-PERSONAL-DATA         GROUP                     0
```

Example 8.2a (cont'd)

```
     URA VERSION 740710                                               JUL 11, 1974   21:38.52
                                         ADS-EXAMPLE
                                   CONSISTS MATRIX REPORT

PARAMETERS FOR: CM

  FILE CONSISTS

   URA315:CONCUL: THE FOLLOWING DO NOT CONSIST OF ANYTHING:

        BAD-INPUT-DATA
        DATE
        EMPLOYEE-DATA
        EXPLICIT-PERSONAL-DATA
        NEW-EMP-VALIDATION
        PAY-UNITS
        RELATED-PERSONAL-DATA
        TAX-UNITS
        VALID-T-CARD
        VALID-TERM-INFO

    **ROW**                                 **COLUMN**

  1 EMPLOYEE-NAME                         1 BAD-INPUT-DATA
    (ELEMENT)                               (GROUP)
  2 NUMBER-OF-DEPENDENTS                  2 DATE
    (ELEMENT)                               (GROUP)
  3 PAYRATE                               3 EMPLOYEE-DATA
    (ELEMENT)                               (GROUP)
  4 EMPLOYEE-NUMBER                       4 EXPLICIT-PERSONAL-DATA
    (ELEMENT)                               (GROUP)
  5 VARYING-EMPLOYEE-DATA                 5 NEW-EMP-VALIDATION
    (ENTITY)                                (GROUP)
  6 TERMINATION-CODE                      6 NEW-EMPLOYEE-PART
    (ELEMENT)                               (GROUP)
  7 NEW-EMPLOYEE-PART                     7 PAY-UNITS
    (GROUP)                                 (SET)
  8 TAX-RATE                              8 PAYROLL-MASTER-INFORMATION
    (ELEMENT)                               (SET)
  9 YTD-DEDUCT                            9 RELATED-PERSONAL-DATA
    (ELEMENT)                               (GROUP)
 10 YTD-GROSS                            10 TAX-UNITS
    (ELEMENT)                               (SET)
                                         11 TERMINATED-EMPLOYEE-PART
```

Example 8.2b

```
URA VERSION 740710                                   JUL 11, 1974   21:38.52
                            ADS-EXAMPLE

                      CONSISTS MATRIX REPORT

                           (GROUP)
                        12 VALID-NEW-INFO
                           (GROUP)
                        13 VALID-T-CARD
                           (GROUP)
                        14 VALID-TERM-INFO
                           (GROUP)
                        15 VARYING-EMPLOYEE-DATA
                           (ENTITY)

            THE ROWS ARE CONTAINED IN THE COLUMNS WITH *S


                             1 11111
                    1234567890 12345
                   +----------+-----+
                 1 I     *    I*    I
                 2 I     *    I    *I
                 3 I     *    I    *I
                 4 I     *    I*    I
                 5 I       *  I     I
                   +----------+-----+
                 6 I          I*    I
                 7 I          I *   I
                 8 I          I    *I
                 9 I          I    *I
                10 I          I    *I
                   +----------+-----+
```

Example 8.2b (cont'd)

```
JRA VERSION 740710                                          JUL 11, 1974   21:38.52
                                   ADS-EXAMPLE

                             CONSISTS MATRIX REPORT

**THE NUMBER OF COLUMNS THAT CONTAIN THE ROWS**

    ROW                             TYPE                       COUNT

 1  EMPLOYEE-NAME                   ELEMENT                        2

 2  NUMBER-OF-DEPENDENTS            ELEMENT                        2

 3  PAYRATE                         ELEMENT                        2

 4  EMPLOYEE-NUMBER                 ELEMENT                        2

 5  VARYING-EMPLOYEE-DATA           ENTITY                         1

 6  TERMINATION-CODE                ELEMENT                        1

 7  NEW-EMPLOYEE-PART               GROUP                          1

 8  TAX-RATE                        ELEMENT                        1

 9  YTD-DEDUCT                      ELEMENT                        1

10  YTD-GROSS                       ELEMENT                        1

**THE NUMBER OF ROWS CONTAINED IN THE COLUMNS**

    COLUMN                          TYPE                       COUNT

15  VARYING-EMPLOYEE-DATA           ENTITY                         5

 6  NEW-EMPLOYEE-PART               GROUP                          4

11  TERMINATED-EMPLOYEE-PART        GROUP                          3

 8  PAYROLL-MASTER-INFORMATION      SET                            1

12  VALID-NEW-INFO                  GROUP                          1
```

Example 8.2b (cont'd)

```
   URA VERSION 740710                                            JUL 11, 1974  21:38.52
                                       ADS-E. ,PLE

                                 CONSISTS MATRIX REPORT

  1 BAD-INPUT-DATA                     GROUP                          0

  2 DATE                               GROUP                          0

  3 EMPLOYEE-DATA                      GROUP                          0

  4 EXPLICIT-PERSONAL-DATA             GROUP                          0

  5 NEW-EMP-VALIDATION                 GROUP                          0

  7 PAY-UNITS                          SET                            0

  9 RELATED-PERSONAL-DATA              GROUP                          0

 10 TAX-UNITS                          SET                            0

 13 VALID-T-CARD                       GROUP                          0

 14 VALID-TERM-INFO                    GROUP                          0
```

Example 8.2b (cont'd)

8.3 CONTENTS (CONT)

Report Description

The CONTENTS REPORT is generated by this command. It presents the data structure specified through usage of the CONSISTS statement in the problem statement. For this reason, only SET, INPUT, OUTPUT, ENTITY and GROUP names may be used as input to the command. All names used as input to the command are designated as level 1 data objects (in the report) and each subordinate level is identified by a level number and the names of all data objects in that level. The lower levels will consist of INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names.

Implementation

To obtain the basic report for one data object, specify:

```
CONTENTS N=VARYING-EMPLOYEE-DATA
```

This is a simple way of finding out all the data which is contained within this one object. If the data structure represented by the report is not satisfactory, more information (in the form of URL statements) may have to be added to make it "complete." It must be remembered that only subordinate levels are presented and any levels above VARYING-EMPLOYEE-DATA (it may be contained in a SET) are not shown. The output generated by this statement is shown in Example 8.3a.

Options and Alternatives

This report can be obtained for several names at one time by using the FILE parameter:

```
EDIT CONT.DATA NONUM NEW
NEW-EMPLOYEE-INFORMATION
PAY-STATEMENT
PAYSYSTEM-OUTPUTS
TIME-CARD
(blank line RETURN)
SE
EXEC CLI
     CONTENTS FILE=CONT.DATA
```

The results of this proceudre are shown in Example 8.3b. The asterisk (*) designates the name as being one of the input names to the command. The numbers beneath the asterisk specify the number of subordinate data objects (with respect to the level 1 object) in the order they were retrieved from the data base to be printed. Note that PAYSYSTEM-OUTPUTS has no subordinate objects (or at least none via the CONSISTS statement). It is possible that a structure may be specified by use of the SUBPARTS statement, also. (See the STRUCTURE command, Section 8.17.)

The rumber of subordinate levels to be printed for each name may be controlled via the LEVELS parameter. If LEVELS=2 we e also giver in the above example for CONTENTS, only the level 1 and level 2 names would appear on the report. It is wise tu utilize this parameter when you have described a large data structure in the problem statement and it is difficult to estimate what the size of the resulting output will be when a CONTENTS command is used.

When the output is thought to be large, however, it is usually a good idea to also specify the INDEX parameter to generate an index into the report.

The command can also perform some "completeness" checks on the information presented in the report by specifying the NCFLAG parameter. This results in having all undefined data objects (those whose name type is *** UNKNOWN or AMBIGUOUS *** flagged and any GROUPS which do not consist of any lower level data objects flagged. (By logical definition, a group consists of one or more objects.)

Usage With NAME-GEN

NAME-GEN is very often used in conjunction with the CONTENTS command:

```
NAME-GEN SET
CONTENTS
```

This sequence of URA command produces a listing of all SET names in the URA data base (via the NAME-GEN command) and then produces a CONTENTS REPORT using each SET name as a level 1 data object. This should give you a fairly complete representation of your problem statement's data structure.

Common Errors

Care must be taken that only SET, INPUT, OUTPUT, ENTITY and GROUP names are used as input to this command. Any other name types will not be allowed. (ELEMENT can not be given because an ELEMENT can not CONSIST of any subordinate data objects by URL/URA definition.)

Cases have occurred where an excessively large output has been produced because the problem definers were unable to estimate the size of the data structure they had specified. Constantly keep this in mind, and when in doubt produce the output a level at a time via the LEVELS parameter.

```
URA VERSION 740710                                      JUL 28, 1974  21:34.54
                           ADS-EXAMPLE

                         CONTENTS REPORT

PARAMETERS FOR: CONT

  NAME=VARYING-EMPLOYEE-DATA NONCFLAG NOINDEX LEVELS=ALL

  1*     1 VARYING-EMPLOYEE-DATA (ENTITY)
  1        2   PAYRATE (ELEMENT)
  2        2   EMPLOYEE-NUMBER (ELEMENT)
  3        2   TAX-RATE (ELEMENT)
  4        2   NUMBER-OF-DEPENDENTS (ELEMENT)
  5        2   YTD-DEDUCT (ELEMENT)
  6        2   YTD-GROSS (ELEMENT)
```

Example 8.3a

```
 JRA VERSION 740328                                              JUL  7, 1974  18:44.24
                                   ADS-EXAMPLE

                                       CONTENTS REPORT

PARAMFTERS FOR: CONT

  FILE NONCFLAG NOINDEX LEVELS=ALL

  1*      1 NEW-EMPLOYEE-INFORMATION (INPUT)
   1       2    EMPLOYEE-NAME (ELEMENT)
   2       2    NUMRER-UF-DEPENDENTS (ELEMENT)
   3       2    PAYRATE (ELEMENT)
   4       2    EMPLOYEE-NUMBER (ELEMENT)

  2*      1 PAY-STATEMENT (OUTPUT)
   1       2    CHECK (GROUP)
   2       2    STUB (GROUP)
   3        3      EMPLOYEE-NUMBER (ELEMENT)
   4        3      PAYRATE (ELEMENT)
   5        3      GROSS-PAY (ELEMENT)
   6        3      NET-PAY (ELEMENT)

  3*      1 PAYSYSTEM-OUTPUTS (OUTPUT)

  4*      1 TIME-CARD (INPUT)
   1       2    EMPLOYEE-NUMBER (ELEMENT)
   2       2    HOURS-WORKED (ELEMENT)
```

Example 8.3b

8.4 DATA-BASE-STATISTICS (DBS)

Report Description

This command generates the DATA BASE STATISTICS report. It presents all names defined in the URA data base along with information related to the physical URA data base. This information is of relatively little interest to the user.
This report is intended for those people implementing and maintaining the URA software at their computer installation.

Implementation

To generate the basic report:

```
DBS
```

will produce the printout shown in Example 8.4.

Options and Alternatives

Additional information may be added to the report by specifying SYNONYM and NAMNUBS as parameters for the DBS command.

Usage with NAME-GEN

This report cannot be used with NAME-GEN.

Common Errors

Since this command is not intended for use by the user (it does not directly help him in defining a user requirement), the output should be generated only for those persons that know how to interpret the information on it.

```
      URA VERSION 740328                                              JUL  7, 1974  18:00.55
                                         ADS-EXAMPLE

                                        DATA BASE STATISTICS

PARAMETERS FOR: DBS

   NAMES NUBS NOSYNONYMS NONAMNUBS

SEQ       NAME                                   NUBA NUBB NUBC  COM  OTH RELA NUBA NUBB NUBC  COM  OTH RELB
  1 COMPANY-PROCEDURES-MANUAL             20     0    0    0    0    0    0    1    0    0    0    0    1
  2 DEPARTMENTS-AND-EMPLOYEES             16     0    0    0    1    0    1    2    0    0    0    0    2
  5 HIGHEST-LEVEL-PROCESS                 10     0    0    0    0    0    0    1    0    0    0    0    1
  7 JOSEPH-ISMITH                         14     0    0    0    1    0    1    1    0    0    0    0    1
  8 LEVEL-1                               10     0    0    0    0    0    0    1    0    0    0    0    1
 12 PAYROLL-MASTER-INFORMATION            19     0    1    0    1    0    2    1    0    0    0    0    1
 13 PAYROLL-PROCESSING                    15     6    0    0    1    0    7    1    1    0    0    0    2
 14 PAYSYSTEM-OUTPUTS                     13     1    0    0    1    0    2    2    0    0    0    0    2
 16 RM-228H-WEST-ENGINEERING-BLDG         11     2    0    0    0    0    2    0    0    0    0    0    0
 17 TARGET-SYSTEM                         10     0    0    0    0    0    0    1    0    0    0    0    1
 18 WALTER-J-RATAJ                        14     4    0    0    1    0    5    1    0    0    0    0    1
 19 WEEKLY-EMPLOYEE-INFORMATION            8     1    0    0    1    0    2    2    0    0    0    0    2

   *** TOTALS                                   14    1    0    7    0   22   14    1    0    0    0   15
```

EXAMPLE 8.4

```
     VERSION 740328                                              JUL  7, 1974   18:00.55
                                   ADS-EXAMPLE

                                   DATA BASE STATISTICS

YPE (RELA,RELB)      TYPE (RELA,RELB)      TYPE (RELA,RELB)      TYPE (RELA,RELB)      TYPE (RELA,RELB)
  1 (    0,    0)      11 (    0,    0)      21 (    0,    0)      31 (    1,    1)      41 (    0,    0)
  2 (    0,    0)      12 (    0,    0)      22 (    3,    3)      32 (    0,    0)      42 (    0,    0)
  3 (    0,    0)      13 (    0,    0)      23 (    2,    2)      33 (    0,    0)      43 (    0,    0)
  4 (    0,    0)      14 (    7,    0)      24 (    0,    0)      34 (    0,    0)      44 (    0,    0)
  5 (    0,    0)      15 (    0,    0)      25 (    0,    0)      35 (    1,    1)      45 (    0,    0)
  6 (    0,    0)      16 (    1,    1)      26 (    0,    0)      36 (    0,    0)      46 (    0,    0)
  7 (    0,    0)      17 (    0,    0)      27 (    1,    1)      37 (    0,    0)      47 (    0,    0)
  8 (    0,    0)      18 (    0,    0)      28 (    4,    4)      38 (    0,    0)      48 (    0,    0)
  9 (    0,    0)      19 (    2,    2)      29 (    0,    0)      39 (    0,    0)      49 (    0,    0)
 10 (    0,    0)      20 (    0,    0)      30 (    0,    0)      40 (    0,    0)      50 (    0,    0)
```

EXAMPLE B.4 (cont'd)

8.5 DATA·PROCESS (DP)

Repo·t Description

The DATA PROCESS REPORT is produced by this command. Two matrices are presented that provide information about PROCESSES and data objects (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS). The first matrix describes the interaction between data objects and PROCESSES (that data which USES, RECEIVED, GENERATED, DERIVED or UPDATED for a particular PROCESS). The data objects are represented by the rows and the PROCESSES by the columns. The type of relation is determined by the entry in the matrix. The second matrix describes the data transfer between PROCESSES. In this matrix, PROCESS names represented by rows DERIVE an UPDATE data which is USED by PROCESS names represented by columns. The existence of such a relationship is specified by an asterisk (*).

Implementation

One simple form of this report can be obtained in the following manner:

```
DP  N=PAYROLL-PROCESSING  PROCESS
```

This would present all the data that interacts with the PROCESS "PAYROLL-PROCESSING" in a matrix format. The second matrix would not be produced, however, See Example 8.5a for the printout generated by this command.

Options and Alternatives

The user may use the FILE parameter to specify input:

```
EDIT DP.DATA NONUM NEW
F-DATA
V-DATA
(blank line RETURN)
SE
EXEC CLI
     DP F=DP.DATA DATA
```

By usage of the DATA parameter, the command retrieves all PROCESSES which interact with the names specified in the input file. Note here that when the DATA parameter is given, all names used as input must be data object names (i.e., SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names).

Whenever the PROCESS parameter is used, all the names used as input to the command must be PROCESS names.

Usage With NAME-GEN

The most common method of using this command is as follows:

```
NG  PROCESS
DP  PROCESS
```

This produces the report using all PROCESS names defined in the data base. Example 8.5b presents an example of this implementation.

Common Errors

The user must keep in mind which types of names are allowed for the PROCESS and DATA parameters.

This report proves to be more valuable (in most cases) when generated for several input names. This command is relatively costly to implement and produces a large amount of output.

The matrices can be very large and sparse, so this command should only be used when many processes are completely defined.

```
     URA  VERSION 740710                                                AUG 12, 1974  20:54.50
                                                   ADS-EXAMPLE

                                               DATA PROCESS REPORT

PARAMETERS FOR: DP

   NAME=PAYPROC PROCESS


   THE ROWS ARE DATA NAMES, THE COLUMNS ARE PROCESS NAMES.


     ROW NAMES                                                            COLUMN NAMES

   1 PAY-STATEMENT                          OUTPUT                     1 PAYROLL-PROCESSING              PROCES
   2 TIME-CARD                              INPUT
   3 ERROR-LISTING                          OUTPUT
   4 TERMINATING-EMPLOYEE-INFO              INPUT
   5 NEW-EMPLOYEE-INFORMATION               INPUT
   6 HIRED-TERMINATED-REPORT                OUTPUT
   7 VARYING-EMPLOYEE-DATA                  ENTITY
   8 FIXED-EMPLOYEE-DATA                    ENTITY
   9 PAYROLL-MASTER-INFORMATION             SET
  10 WEEKLY-EMPLOYEE-INFORMATION            INPUT
  11 PAYSYSTEM-OUTPUTS                      OUTPUT
```

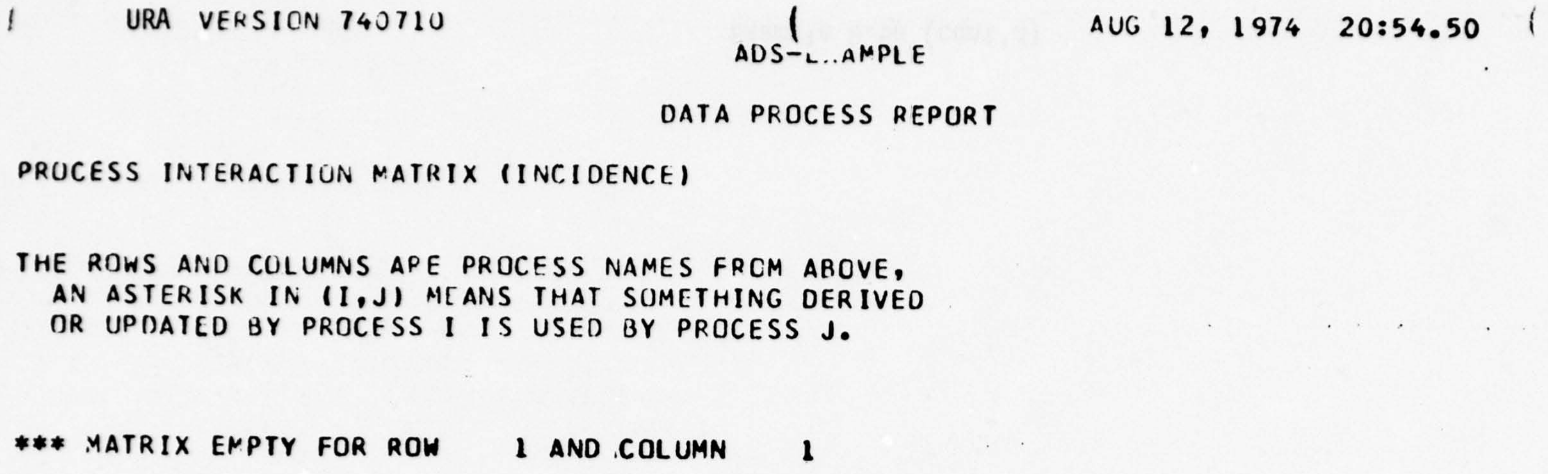
Example 8.5a

```
URA VERSION 740710                                          AUG 12, 1974   20:54.50
                              ADS-EXAMPLE

                          DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX

(I,J) VALUE   MEANING
-----------   -----------------------

    I         ROW I IS INPUT TO COLUMN J
     U        ROW I IS UPDATED BY COLUMN J
      O       ROW I IS OUTPUT OF COLUMN J


                              1
                           +-----+
                         1 I   OI
                         2 I I  I
                         3 I   OI
                         4 I I  I
                         5 I I  I
                           +-----+
                         6 I   OI
                         7 I  U I
                         8 I  U I
                         9 I  U I
                        10 I I  I
                           +-----+
                        11 I   OI
                           +-----+
```

Example 8.5a (cont'd)

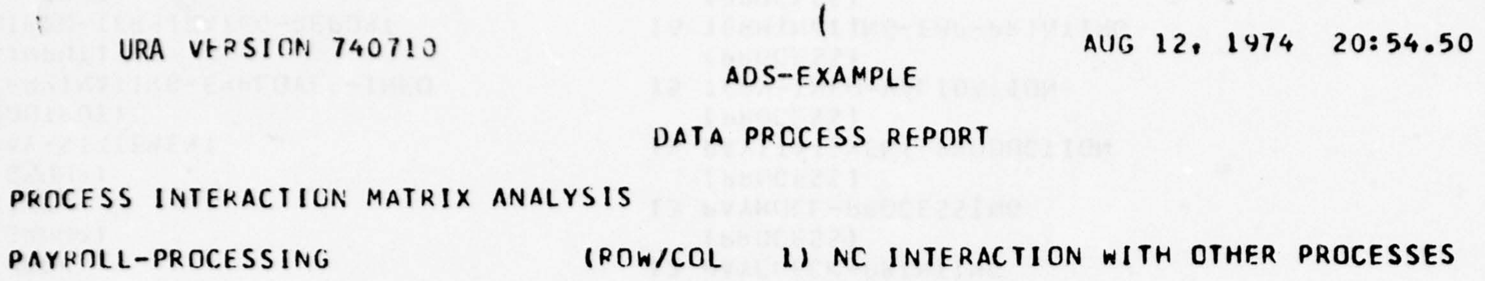
```
    URA VERSION 740710                                                 AUG 12, 1974  20:54.50
                                             ADS-EXAMPLE

                                         DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX ANALYSIS

PAY-STATEMENT                           (ROW    1) NOT GENERATED BY ANY PROCESS
TIME-CARD                               (ROW    2) NOT RECEIVED BY ANY PROCESS
ERROR-LISTING                           (ROW    3) NOT GENERATED BY ANY PROCESS
TERMINATING-EMPLOYEE-INFO               (ROW    4) NOT RECEIVED BY ANY PROCESS
NEW-EMPLOYEE-INFORMATION                (ROW    5) NOT RECEIVED BY ANY PROCESS
HIRED-TERMINATED-REPORT                 (ROW    6) NOT GENERATED BY ANY PROCESS
VARYING-EMPLOYEE-DATA                   (ROW    7) NOT DERIVED BY ANY PROCESS
VARYING-EMPLOYEE-DATA                   (ROW    7) UPDATED, BUT NOT USED BY ANY PROCESS
FIXED-EMPLOYEE-DATA                     (ROW    8) NOT DERIVED BY ANY PROCESS
FIXED-EMPLOYEE-DATA                     (ROW    8) UPDATED, BUT NOT USED BY ANY PROCESS
PAYROLL-MASTER-INFORMATION              (ROW    9) NOT DERIVED BY ANY PROCESS
PAYROLL-MASTER-INFORMATION              (ROW    9) UPDATED, BUT NOT USED BY ANY PROCESS
WEEKLY-EMPLOYEE-INFORMATION             (ROW   10) NOT USED BY ANY PROCESS
PAYSYSTEM-OUTPUTS                       (ROW   11) NOT DERIVED BY ANY PROCESS
```

Example 8.5a (cont'd)



```
URA VERSION 740710                          ADS-L..AMPLE                   AUG 12, 1974   20:54.50

                                       DATA PROCESS REPORT

PROCESS INTERACTION MATRIX (INCIDENCE)

THE ROWS AND COLUMNS APE PROCESS NAMES FROM ABOVE,
  AN ASTERISK IN (I,J) MEANS THAT SOMETHING DERIVED
  OR UPDATED BY PROCESS I IS USED BY PROCESS J.


*** MATRIX EMPTY FOR ROW     1 AND COLUMN     1
```

Example 8.5a (cont'd)



```
URA VFRSION 740710                                               AUG 12, 1974   20:54.50
                                      ADS-EXAMPLE
                                  DATA PROCESS REPORT
PROCESS INTERACTION MATRIX ANALYSIS
PAYROLL-PROCESSING                (ROW/COL   1) NC INTERACTION WITH OTHER PROCESSES
```

Example 8.5a (cont'd)

```
      URA VERSION 740328                                          JUL  6, 1974  16:52.25
                                    ADS-EXAMPLE

                                    DATA PROCESS REPORT

PARAMETERS FOR: DP

   FILE PROCESS


   THE ROWS ARE DATA NAMES. THE COLUMNS ARE PROCESS NAMES.

    **ROW**                                 **COLUMN**

   1 ERROR-LISTING                          1 ERROR-LISTING-PRODUCTION
     (OUTPUT)                                 (PROCESS)
   2 BAD-INPUT-DATA                         2 FIELD-CHECK-NEW
     (GROUP)                                  (PROCESS)
   3 NEW-EMPLOYEE-PART                      3 FIELD-CHECK-PAYCALC
     (GROUP)                                  (PROCESS)
   4 VALID-NEW-INFO                         4 FIELD-CHECK-TERM
     (GROUP)                                  (PROCESS)
   5 NEW-EMPLOYEE-INFORMATION               5 FILE-REFERENCING
     (INPUT)                                  (PROCESS)
   6 PAYROLL-MASTER-INFORMATION             6 NEW-EMPLOYEE-PRINTING
     (SET)                                    (PROCESS)
   7 EMPLOYEE-DATA                          7 NEW-EMPLOYEE-PROCESSING
     (*** UNKNOWN OR AMBIGUOUS ***)           (PROCESS)
   8 VARYING-EMPLOYEE-DATA                  8 NEW-EMPLOYEE-UPDATING
     (ENTITY)                                 (PROCESS)
   9 FIXED-EMPLOYEE-DATA                    9 NEW-INFO-VALIDATION
     (ENTITY)                                 (PROCESS)
  10 VALID-T-CARD                          10 PAYCALC-INPUT-VALIDATION
     (*** UNKNOWN OR AMBIGUOUS ***)           (PROCESS)
  11 TIME-CARD                             11 PAYCALC-UPDATING
     (INPUT)                                  (PROCESS)
  12 CHECK                                 12 PAYCHECK-PRINTING
     (GROUP)                                  (PROCESS)
  13 STUB                                  13 PAYROLL-PROCESSING
     (GROUP)                                  (PROCESS)
  14 PAY-STATEMENT                         14 PAYSTATEMENT-PRODUCTION
     (OUTPUT)                                 (PROCESS)
  15 TERMINATING-EMPLOYEE-INFO             15 TERM-INFO-VALIDATION
     (INPUT)                                  (PROCESS)
  16 HIRED-TERMINATED-REPORT               16 TERMINATING-EMP-PRINTING
     (OUTPUT)                                 (PROCESS)
```

Example 8.6b

```
     URA VERSION 740328                                                         JUL   6, 1974   16:52.25
                                                ADS-EXAMPLE

                                             DATA PROCESS REPORT

  THE ROWS ARE DATA NAMES. THE COLUMNS ARE PROCESS NAMES.

17 WEEKLY-EMPLOYEE-INFORMATION                    17 TERMINATING-EMP-PROCESSING
   (INPUT)                                           (PROCESS)
18 PAYSYSTEM-OUTPUTS                              18 TERMINATING-EMP-UPDATING
   (OUTPUT)                                          (PROCESS)
19 VALID-TERM-INFO
   (*** UNKNOWN OR AMBIGUOUS ***)
20 TERMINATED-EMPLOYEE-PART
   (GROUP)
```

Example 8.5b (cont'd)

```
URA VERSION 740328                                              JUL  6, 1974   16:52.25
                                        ADS-EXAMPLE

                                        DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX

(I,J) VALUE   MEANING
-----------   ------------------------

     I        ROW I IS INPUT TO COLUMN J
      U       ROW I IS UPDATED BY COLUMN J
       O      ROW I IS OUTPUT OF COLUMN J


                                            1    1   1   1   1   1    1   1   1
       1   2   3   4   5    6   7   8   9   0    1   2   3   4   5    6   7   8
   +--------------------+--------------------+--------------------+------------+
 1 I   O                I                    I           O        I            I
 2 I I                  I               O   OI                   OI            I
 3 I                    I   O   O            I                    I            I
 4 I                    I I       I     O    I                    I            I
 5 I                    I     I       I      I         I          I            I
   +--------------------+--------------------+--------------------+------------+
 6 I                    I      U   U         I  U       U   U     I      U   U I
 7 I                    I           O        I I                  I            I
 8 I                    I         I          I          U I       I            I
 9 I                    I         I          I          U I       I            I
10 I                    I                   OI I                  I            I
   +--------------------+--------------------+--------------------+------------+
11 I                    I                 I  I         I  I       I            I
12 I                    I                    I   O I         O    I            I
13 I                    I                    I   O I         O    I            I
14 I                    I                    I       O   O   O    I            I
15 I                    I                    I         I       I  I     I      I
   +--------------------+--------------------+--------------------+------------+
16 I                    I                    I           O        I            I
17 I                    I                    I         I          I            I
18 I                    I                    I           O        I            I
19 I                    I                    I                   OI I          I
20 I                    I                    I                    I   O   O    I
   +--------------------+--------------------+--------------------+------------+
```

Example 8.5b (cont'd)

```
URA VFRSION 740328                                          JUL  6, 1974  16:52.25
                                   ADS-EXAMPLE

                                   DATA PROCESS RFPORT

DATA PROCESS INTERACTION ANALYSIS

PAYROLL-MASTER-INFORMATION      (ROW     6) NOT DERIVED BY ANY PROCESS
PAYROLL-MASTER-INFORMATION      (ROW     6) UPDATED, BUT NOT USED BY ANY PROCESS
VARYING-EMPLOYEE-DATA           (ROW     8) NOT DERIVED BY ANY PROCESS
FIXED-EMPLOYEE-DATA             (ROW     9) NOT DERIVED BY ANY PROCESS
HIRED-TERMINATED-REPURT         (ROW    16) NOT GENERATED BY ANY PROCESS
WEEKLY-EMPLOYEE-INFORMATION     (ROW    17) NOT USED BY ANY PROCESS
PAYSYSTEM-OUTPUTS               (ROW    18) NOT DERIVED BY ANY PROCESS

FIELD-CHECK-NEW                 (COLUMN   2) DOES NOT INTERACT WITH ANY DATA
FIELD-CHECK-PAYCALC             (COLUMN   3) DOES NOT INTERACT WITH ANY DATA
FIELD-CHECK-TERM                (COLUMN   4) DOES NOT INTERACT WITH ANY DATA
FILE-REFERENCING                (COLUMN   5) DOES NOT INTERACT WITH ANY DATA
TERMINATING-EMP-UPDATING        (COLUMN  18) UPDATES SOMETHING, BUT DOES NOT USE ANYTHING
```

Example 8.5b (cont'd)

```
URA VERSION 740323                                   JUL  6, 1974  16:52.25
                              ADS-EXAMPLE

                              DATA PROCESS REPORT

PROCESS INTERACTION MATRIX (INCIDENCE)

THE ROWS AND COLUMNS ARE PROCESS NAMES FROM ABOVE,
  AN ASTERISK IN (I,J) MEANS THAT SOMETHING DERIVED
  OR UPDATED BY PROCESS I IS USED BY PROCESS J.


                  1 11111111
         1234567890 12345678
        +----------+--------+
      1 I          I        I
      2 I          I        I
      3 I          I        I
      4 I          I        I
      5 I          I        I
        +----------+--------+
      6 I          I        I
      7 I          I        I
      8 I          I*       I
      9 I*    * *  I        I
     10 I*         I*       I
        +----------+--------+
     11 I          I *      I
     12 I          I        I
     13 I       *  I   *    I
     14 I          I *      I
     15 I*         I     *  I
        +----------+--------+
     16 I          I        I
     17 I          I        I
     18 I          I        I
        +----------+--------+
```

Example 8.5b (cont'd)

```
URA VERSION 740328                                              JUL  6, 1974  16:52.25
                              ADS-EXAMPLE

                              DATA PROCESS REPORT

PROCESS INTERACTION MATRIX ANALYSIS

ERROR-LISTING-PRODUCTION          (ROW/COL   1) NO SUCCESSESORS FOR THIS PROCESS
FIELD-CHECK-NEW                   (ROW/COL   2) NO INTERACTION WITH OTHER PROCESSES
FIELD-CHECK-PAYCALC               (ROW/COL   3) NO INTERACTION WITH OTHER PROCESSES
FIELD-CHECK-TERM                  (ROW/COL   4) NO INTERACTION WITH OTHER PROCESSES
FILE-REFERENCING                  (ROW/COL   5) NO INTERACTION WITH OTHER PROCESSES
NEW-EMPLOYEE-PRINTING             (ROW/COL   6) NO SUCCESSESORS FOR THIS PROCESS
NEW-EMPLOYEE-PROCESSING           (ROW/COL   7) NO INTERACTION WITH OTHER PROCESSES
NEW-INFO-VALIDATION               (ROW/COL   9) NO PREDESSESORS FOR THIS PROCESS
PAYCALC-INPUT-VALIDATION          (ROW/COL  10) NO PREDESSESORS FOR THIS PROCESS
PAYCHECK-PRINTING                 (ROW/COL  12) NO SUCCESSESORS FOR THIS PROCESS
PAYROLL-PROCESSING                (ROW/COL  13) NO PREDESSESORS FOR THIS PROCESS
TERM-INFO-VALIDATION              (ROW/COL  15) NO PREDESSESORS FOR THIS PROCESS
TERMINATING-EMP-PRINTING          (ROW/COL  16) NO SUCCESSESORS FOR THIS PROCESS
TERMINATING-EMP-PROCESSING        (ROW/COL  17) NO INTERACTION WITH OTHER PROCESSES
TERMINATING-EMP-UPDATING          (ROW/COL  18) NO INTERACTION WITH OTHER PROCESSES
```

Example 8.5b (cont'd)

8.6 DICTIONARY (DICT)

Report Description

The output from this command is called the DICTIONARY REPORT. For each name used as input to the command the report presents description and identification information. Any type of name (ATTRIBUTE, PROCESS, MAILBOX, etc.) may have this information presented by the DICTIONARY REPORT. More specifically, this information is derived from the DESCRIPTION, SYNONYM, KEYWORDS and RESPONSIBLE-PD statements in the URA data base for each of the names used as input. Its name type is also presented as part of the identification information.

Implementation

To produce the DICTIONARY REPORT for one name only, specify:

```
DICTIONARY  N=PAYROLL-PROCESSING
```

The output produced from this command is shown in Example 8.6a. Even if some of the description or identification information that is to be retrieved by this command has not been specified for this name, the information that has been specified will be presented.

Options and Alternatives

Any of the information retrieved for this report (DESCRIPTION, KEYWORDS, etc.) can be omitted by prefixing the parameter with "NO." For example,

```
DICT  N=PAYROLL-PROCESSING  NODESCRIPTION
```

would present the same information as shown in Example 8.6a, but without the DESCRIPTION statement.

The report may be generated for several names by using the FILE parameter:

```
EDIT DICT.DATA NONUM NEW
CHECK
STUB
(blank line RETURN)
SE
EXEC CLI
     DICT FILE=DICT.DATA
```

In which the report would be generated for the two names, CHECK and STUB. When the report is being produced for several names, the user may make the printout more readable by assigning the NUM-SPACE parameter with some value larger than 2. This parameter determines the number of spaces to be inserted between any two entries on the output.

Usage with NAME-GEN

It is a trivial matter to generate a "complete" data dictionary when the DICTIONARY command is used together with NAME-GEN:

```
NG  GROUP  ELEMENT
DICTIONARY
```

This sequence produces the DICTIONARY report for all GROUP and ELEMENT names that have been defined in the user requirement , which proves to be an excellent reference in later work.

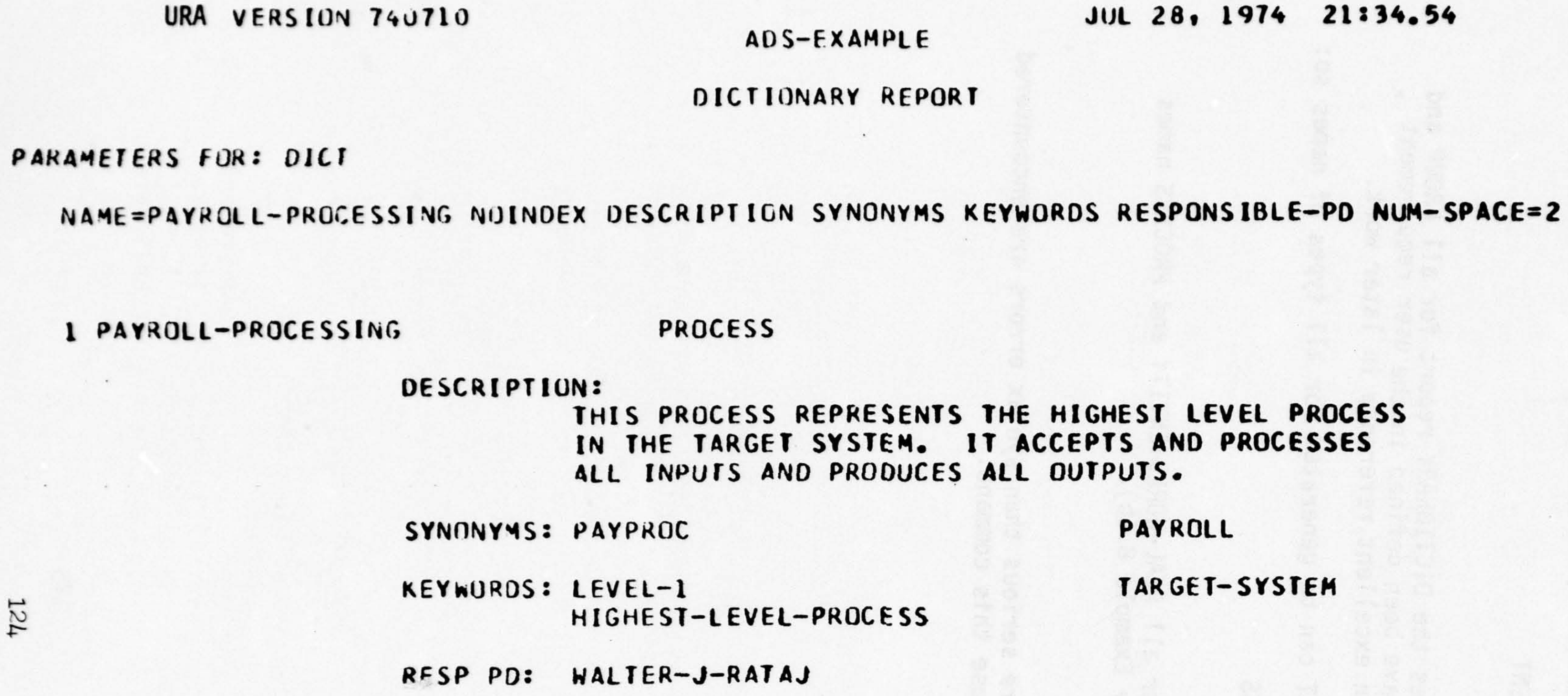
The DICTIONARY REPORT can be generated for all types of names so:

```
NG  RWE  PROCESS
DICT
```

produces a report for all REAL-WORLD-ENTITY and PROCESS names defined (as shown by Example 8.6b).

Common Errors

Usually, nothing more serious than syntax errors are encountered when attempting to use this command.

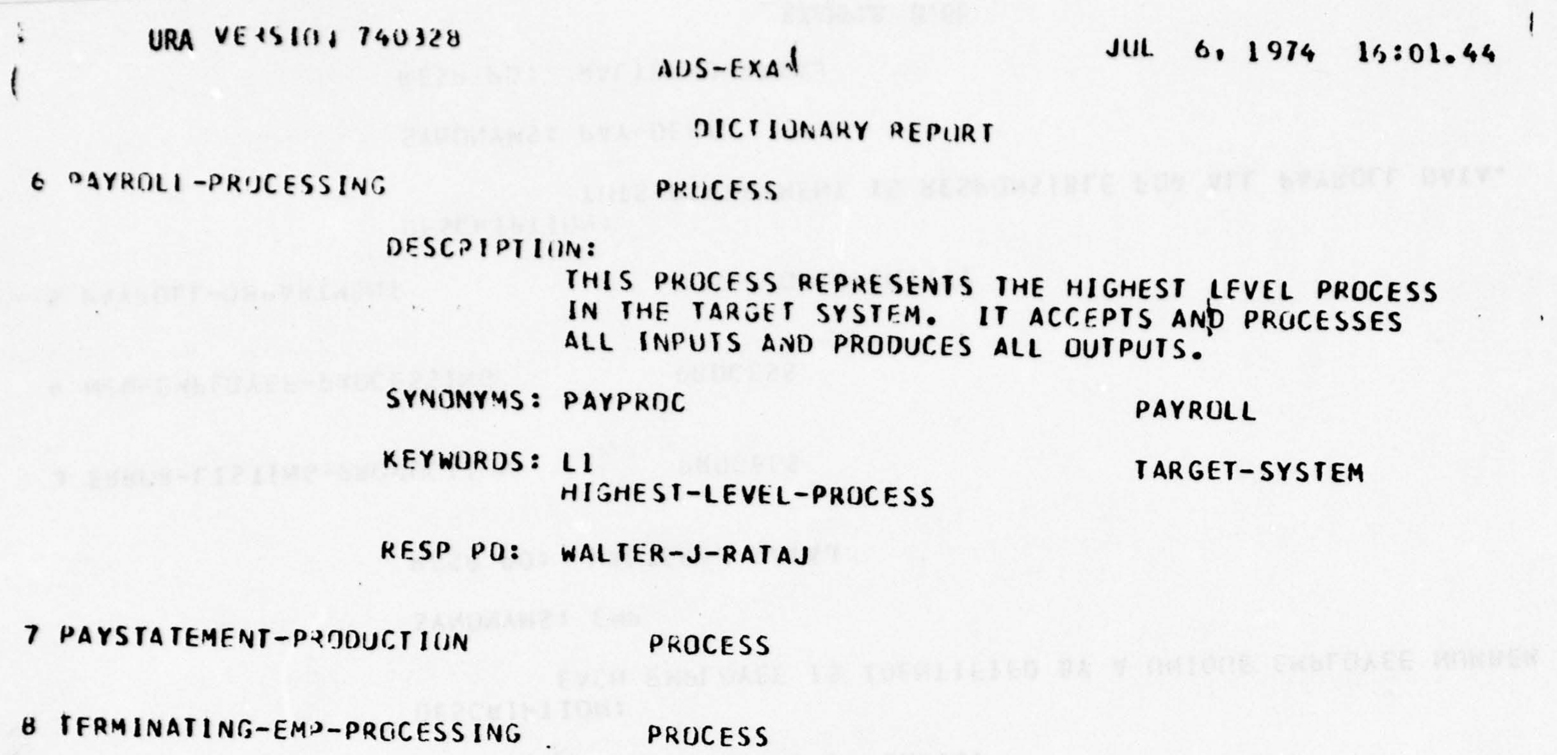


```
    URA  VERSION 740710                                         JUL 28, 1974   21:34.54
                                        ADS-EXAMPLE

                                     DICTIONARY REPORT

PARAMETERS FOR: DICT

   NAME=PAYROLL-PROCESSING NOINDEX DESCRIPTION SYNONYMS KEYWORDS RESPONSIBLE-PD NUM-SPACE=2


   1 PAYROLL-PROCESSING                     PROCESS

                          DESCRIPTION:
                                   THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
                                   IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
                                   ALL INPUTS AND PRODUCES ALL OUTPUTS.

                          SYNONYMS: PAYPROC                             PAYROLL

                          KEYWORDS: LEVEL-1                             TARGET-SYSTEM
                                    HIGHEST-LEVEL-PROCESS

                          RESP PD:  WALTER-J-RATAJ
```

Example 8.6a

```
     URA VERSION 740328                                     JUL   6, 1974   16:01.44
                                  ADS-EXAMPLE
                                         DICTIONARY REPORT

PARAMETERS FOR: DICT

   FILE NOINDEX DESCRIPTION SYNONYMS KEYWORDS RESPONSIBLE-PD NUM-SPACE=2


   1 DEPARTMENTS-AND-EMPLOYEES          REAL-WORLD-ENTITY

                         DESCRIPTION:
                                 THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND
                             SUPPLY ALL THE INPUTS.

                         SYNONYMS: DEPT-EMP


   2 EMPLOYEE                           REAL-WORLD-ENTITY

                         DESCRIPTION:
                               EACH EMPLOYEE IS IDENTIFIED BY A UNIQUE EMPLOYEE NUMBER

                         SYNONYMS: EMP

                         RESP PD:  WALTER-J-RATAJ


   3 ERROR-LISTING-PRODUCTION           PROCESS


   4 NEW-EMPLOYEE-PROCESSING            PROCESS


   5 PAYROLL-DEPARTMENT                 REAL-WORLD-ENTITY

                         DESCRIPTION:
                                 THIS DEPARTMENT IS RESPONSIBLE FOR ALL PAYROLL DATA.

                         SYNONYMS: PAY-DEPT

                         RESP PD:  WALTER-J-RATAJ
```

EXAMPLE 8.6b



```
      URA VERSION 740328                                              JUL  6, 1974  16:01.44
                               ADS-EXA
                                         DICTIONARY REPORT
   6 PAYROLL-PROCESSING                 PROCESS
                        DESCRIPTION:
                                THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
                                IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
                                ALL INPUTS AND PRODUCES ALL OUTPUTS.
                        SYNONYMS: PAYPROC                                   PAYROLL
                        KEYWORDS: L1                                        TARGET-SYSTEM
                                  HIGHEST-LEVEL-PROCESS
                        RESP PD:  WALTER-J-RATAJ

   7 PAYSTATEMENT-PRODUCTION            PROCESS

   8 TERMINATING-EMP-PROCESSING         PROCESS
```

EXAMPLE 8.6b (cont'd)

8.7 ENTITY-IDENTIFIER (EI)

Report Description

The IDENTIFIER INFORMATION REPORT is generated by this command. It presents a matrix where IDENTIFIER names are represented by the rows in the matrix and ENTITIES by the columns. The existence of an IDENTIFIES relationship between IDENTIFIER and ENTITY is specified by an asterisk. A summary is also produced to aid in describing the information portrayed in the matrix.

Implementation

One simple form of this command is to retrieve all the ENTITIES that a particular IDENTIFIER "IDENTIFIES":

```
EI  N=EMPLOYEE-NUMBER  I
```

Whenever the "I" (IDENTIFIER) parameter is used, all names used as input to the command must be IDENTIFIER names:

Options and Alternatives:

To generate the report for several names, the "FILE" parameter can be used:

```
EDIT EI.DATA NONUM NEW
F-DATA
V-DATA
(blank line RETURN)
SE
EXEC CLI
     EI FILE=EI.DATA ENTITY
```

Here, the names used as input are ENTITY names and the command will retrieve all those IDENTIFIERS which identify the ENTITIES. The report generated from this procedure is given in Example 7.7a.

Usage with NAME-GEN

To retrieve all IDENTIFIERS which are also GROUPS and all IDENTIFIERS which are also ELEMENTS to be used as input to the command:

```
NG  IDG  IDE

EI  I
```

The report will be produced using all the IDENTIFIER (which are also GROUPS and ELEMENTS) in the data base. Another common form of using NAME-GEN in conjunction with the command is shown below.

```
NG  ENTITY

EI  E
```

The report will be produced using all the IDENTIFIERS (which are also GROUPS and ELEMENTS) in the data base. Another common form of using NAME-GEN in conjunction with the command is shown below.

```
NG ENTITY
EI  E
```

The report will be generated using all the ENTITIES defined in the URA data base. For our particular user requirement , the output produced from this sequence of commands is the same as shown in Example 7.7a.

Common Errors

Since not that many ENTITIES or IDENTITIES are defined in an average user requirement , the only thing to watch out for is that when ENTITIES are used as input the ENTITY parameter must be given, etc.

```
     URA  VERSION 740710                                                 JUL 11, 1974   16:18.20
                                                ADS-EXAMPLE

                                       IDENTIFIER INFORMATION REPORT

PARAMETERS FOR: FI

   FILE ENTITY

    **ROW**                                      **COLUMN**

  1 EMPLOYEE-NUMBER                                1 FIXED-EMPLOYEE-DATA
    (ELEMENT)                                        (ENTITY)
                                                   2 VARYING-EMPLOYEE-DATA
                                                     (ENTITY)

                                   THE ROWS ARE IDENTIFIERS OF THE COLUMNS WITH *S



                                                    12
                                                   +--+
                                                 1 I**I
                                                   +--+
```

Example 8.7a

```
     URA VERSION 740710                                                  JUL 11, 1974  16:18.20
                                                  ADS-EXAMPLE

                                         IDENTIFIER INFORMATION REPORT

**THE NUMBER CF COLUMNS IDENTIFIED BY THE ROWS**

    ROW                                   TYPE                                  COUNT

 1  EMPLOYEE-NUMBER                       ELEMENT                                   2

**THE NUMBER OF ROWS THAT IDENTIFY THE COLUMNS**

    COLUMN                                TYPE                                  COUNT

 1  FIXED-EMPLOYEE-DATA                   ENTITY                                    1

 2  VARYING-EMPLCYEE-DATA                 ENTITY                                    1
```

Example 8.7a (con'd)

8.8 FORMATTED-PROBLEM-STATEMENT (FPS)

Report Description

The FORMATTED PROBLEM STATEMENT is generated when this command is implemented. Any type of name in the data base can be used as input to this command. For each name used, the command retrieves all information specified for that name in the data base and presents this in the FORMATTED PROBLEM STATEMENT. The report is generated in sections. The first statement in each section presents the name and name type being described and the remainder of the section consists of statements to describe this name.

Implementation

To generate an FPS report for one name, the format is:

```
FPS  N=FIELD-CHECK-NEW
```

All the information known about FIELD-CHECK-NEW in the URA data base will be retrieved and formatted as shown by Example 8.8a.

Options and Alternatives

The format of the information presented by this report can be varied tremendously by using the format parameters:

```
SMARG
NMARG
AMARG
BMARG
RNMARG
CMARG
HMARG
ONE-PER-LINE
NEW-PAGE
NEW-LINE
```

Figure 8a illustrates most of the margin parameters by presenting their default values. (HMARG has the value 40 unless specified otherwise.) A complete description for each of these parameters can be found in Part II .

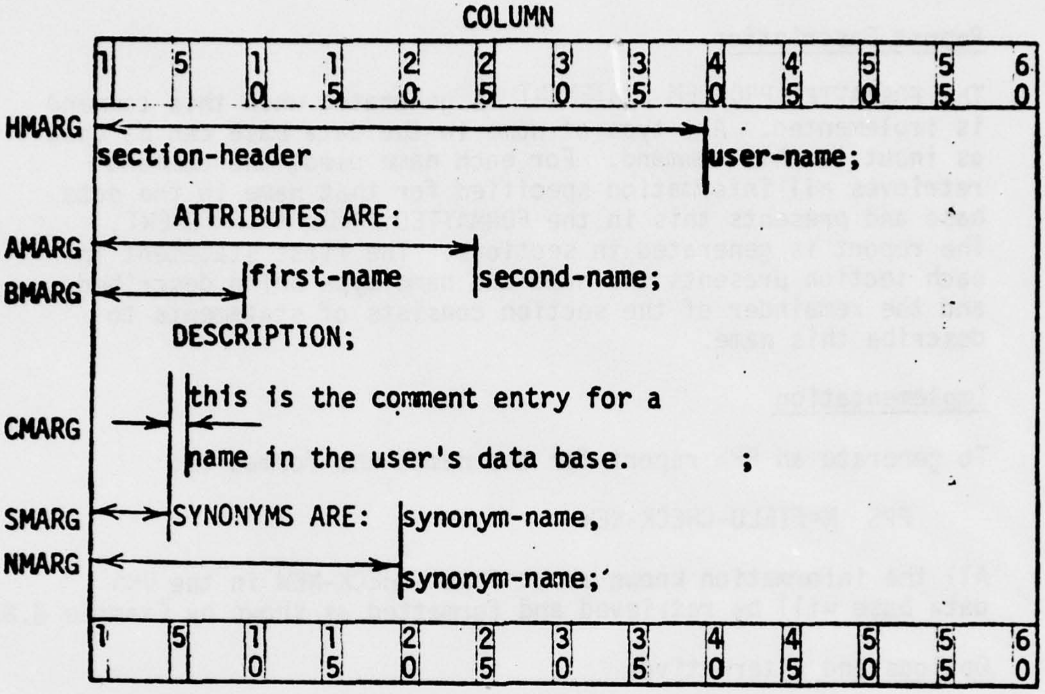


Figure 8a

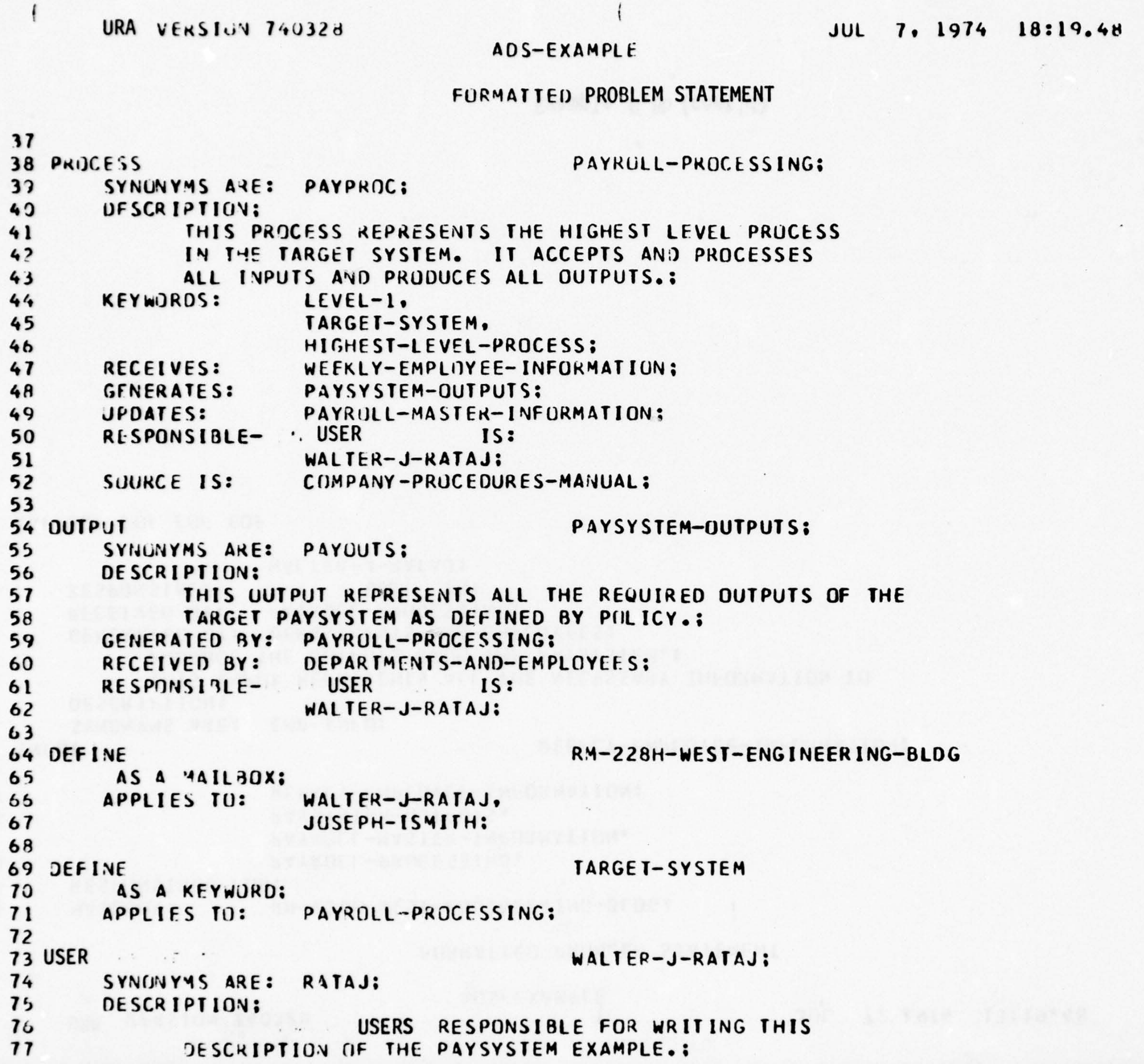
Another valuable parameter for this command is PUNCH. It specifies that all the information presented in the FORMATTED PROBLEM STATEMENT should also be written into a PUNCH file, but with a format acceptable by the INPUT-PSL command. This would be used if a copy of the user requirement was to be sent to another installation. In this case, the PUNCH information might be put on cards or on tape.

```
     URA VERSION 740328                                          JUN  8, 1974  14:04.08

                                       FORMATTED PROBLEM STATEMENT

PARAMETERS FOR: FPS

   NAME=FIELD-CHECK-NEW NOINDEX PRINT PUNCH SMARG=5 NMARG=20 AMARG=10 BMARG=25 RNMARG=70 CMARG=1
   HMARG=4C DESG ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE

   1 PROCESS                                      FIELC-CHECK-NEW;
   2     PART CF:         NEW-EMPLOYEE-UPDATING;
   3     UTILIZFD BY:     NEW-INFO-VALIDATION;
   4
   5 EOF EOF EOF EOF EOF
```

Example 8.8a

```
URA VERSION 740323                                                              JUL  7, 1974  13:19.48
                                        ADS-EXAMPLE

                                  FORMATTED PROBLEM STATEMENT

PARAMETERS FOR: FPS

   FILE NOINDEX PRINT NOPUNCH SMARG=5 NMARG=20 AMARG=10 BMARG=25 RNMARG=70 CMARG=1 HMARG=40 DESG
   ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE

    1 DEFINE                                         COMPANY-PROCEDURES-MANUAL
    2      AS A SOURCE:
    3     APPLIES TO:     PAYROLL-PROCESSING;
    4
    5 REAL-WORLD-ENTITY                              DEPARTMENTS-AND-EMPLOYEES;
    6     SYNONYMS ARE:   DEPT-EMP;
    7     DESCRIPTION;
    8           THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND
    9       SUPPLY ALL THE INPUTS.;
   10     GENERATES:      WEEKLY-EMPLOYEE-INFORMATION;
   11     RECEIVES:       PAYSYSTEM-OUTPUTS;
   12
   13 DEFINE                                         HIGHEST-LEVEL-PROCESS
   14      AS A KEYWORD:
   15     APPLIES TO:     PAYROLL-PROCESSING;
   16
   17 USER                                           JOSEPH-ISMITH;
   18     SYNONYMS ARE:   JI;
   19     DESCRIPTION;
   20                         USERS RESPONSIBLE FOR WRITING THIS
   21           DESCRIPTION OF THE PAYSYSTEM EXAMPLE.;
   22     MAILBOX:        RM-228H-WEST-ENGINEERING-BLDG;
   23
   24 DEFINE                                         LEVEL-1
   25      AS A KEYWORD;
   26     APPLIES TO:     PAYROLL-PROCESSING;
   27
   28 SET                                            PAYROLL-MASTER-INFORMATION;
   29     SYNONYMS ARE:   PAY-MAST;
   30     DESCRIPTION;
   31           THIS SET CONTAINS ONE UNIT OF INFORMATION
   32           FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,
   33           THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;
   34     UPDATED BY:     PAYROLL-PROCESSING;
   35     RESPONSIBLE- USER              IS:
   36                     WALTER-J-RATAJ;
```

Example 8.8b



```
     URA VERSION 740328                                        JUL  7, 1974  18:19.48
                                        ADS-EXAMPLE

                                  FORMATTED PROBLEM STATEMENT

37
38 PROCESS                                             PAYROLL-PROCESSING;
39     SYNONYMS ARE:   PAYPROC;
40     DESCRIPTION;
41          THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
42          IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
43          ALL INPUTS AND PRODUCES ALL OUTPUTS.;
44     KEYWORDS:       LEVEL-1,
45                     TARGET-SYSTEM,
46                     HIGHEST-LEVEL-PROCESS;
47     RECEIVES:       WEEKLY-EMPLOYEE-INFORMATION;
48     GENERATES:      PAYSYSTEM-OUTPUTS;
49     UPDATES:        PAYROLL-MASTER-INFORMATION;
50     RESPONSIBLE-     USER          IS:
51                     WALTER-J-RATAJ;
52     SOURCE IS:      COMPANY-PROCEDURES-MANUAL;
53
54 OUTPUT                                              PAYSYSTEM-OUTPUTS;
55     SYNONYMS ARE:   PAYOUTS;
56     DESCRIPTION;
57          THIS OUTPUT REPRESENTS ALL THE REQUIRED OUTPUTS OF THE
58          TARGET PAYSYSTEM AS DEFINED BY POLICY.;
59     GENERATED BY:   PAYROLL-PROCESSING;
60     RECEIVED BY:    DEPARTMENTS-AND-EMPLOYEES;
61     RESPONSIBLE-      USER         IS:
62                     WALTER-J-RATAJ;
63
64 DEFINE                                              RM-228H-WEST-ENGINEERING-BLDG
65       AS A MAILBOX;
66     APPLIES TO:     WALTER-J-RATAJ,
67                     JOSEPH-ISMITH;
68
69 DEFINE                                              TARGET-SYSTEM
70       AS A KEYWORD;
71     APPLIES TO:     PAYROLL-PROCESSING;
72
73 USER                                                WALTER-J-RATAJ;
74     SYNONYMS ARE:   RATAJ;
75     DESCRIPTION;
76                          USERS  RESPONSIBLE FOR WRITING THIS
77          DESCRIPTION OF THE PAYSYSTEM EXAMPLE.;
```

Example 8.8b (cont'd)

```
  URA VERSION 740328                                       JUL  7, 1974  18:19.48
                                  ADS-EXAMPLE

                          FORMATTED PROBLEM STATEMENT

78     MAILBOX:      RM-228H-WEST-ENGINEERING-BLDG;
79     RESPONSIBLE FOR:
80                   PAYROLL-PROCESSING,
81                   PAYROLL-MASTER-INFORMATION,
82                   PAYSYSTEM-OUTPUTS,
83                   WEEKLY-EMPLOYEE-INFORMATION;
84
85 INPUT                                  WEEKLY-EMPLOYEE-INFORMATION;
86     SYNONYMS ARE:  EMP-INFO;
87     DESCRIPTION;
88          THIS INPUT REPRESENTS ALL THE NECESSARY INFORMATION TO
89          PRODUCE THE OUTPUTS FROM THE PAYSYSTEM.;
90     GENERATED BY:  DEPARTMENTS-AND-EMPLOYEES;
91     RECEIVED BY:   PAYROLL-PROCESSING;
92     RESPONSIBLE-        USER   IS:
93                   WALTER-J-RATAJ;
94
95 EOF EOF EOF EOF EOF
```

Example 8.8b (cont'd)

Of course the FILE parameter is allowed by this command, and so the FPS may be generated for several names:

```
EDIT FPS.DATA NONUM NEW
PAY-STATEMENT
ERROR-LIST
HIRED-TERMINATED-REPORT
(blank line RETURN)
SE
EXEC CLI
     FPS FILE=FPS.DATA
```

Usage with NAME-GEN

The FPS is often generated for all names in the data base to check the status of the problem statement in terms of size, consistency, etc. To do this:

```
NG  ALL

FPS
```

Example 8.8b presents the output generated by this sequence of commands for a small user requirement .

Common Errors

It must be kept in mind that the volume of information presented by the FPS command is several times larger than the volume of information entered via INPUT-URL. For this reason, care must be taken when generating an FPS for large user requirements because it can be very costly.

8.9 FREQUENCY (FREQ)

Report Description

This report presents information associated with the HAPPENS statement for all INPUT, OUTPUT, EVENT and PROCESS names in the data base. The report is broken into sections, with each section presenting the frequency of several objects for a particular interval.

Implementation

The only way to generate the output is to specify:

```
FREQUENCY
```

Example 8.9 presents a simple form of the report for one interval.

Options and Alternatives

None

Usage with NAME-GEN

Not applicable

Common Errors

```
URA VERSION 740710                                           JUL 11, 1974  21:38.52
                               ADS-EXAMPLE

                    U R A   F R E Q U E N C Y   R E P O R T

I N T E R V A L :  CALENDAR-WEEK

    N A M E                         T Y P E     T I M E S   H A P P E N S
    OCCURRENCE-OF-BAD-INPUT         EVENT       NUMBER-OF-BAD-INPUTS
    TERMINATING-EMPLOYEE-INFO       INPUT       1
    NEW-EMPLOYEE-INFORMATION        INPUT       1
    TIME-CARD                       INPUT       1
    PAY-STATEMENT                   OUTPUT      NUMBER-OF-EMPLOYEES
```

Example 8.9

8.10 KWIC

Report Description

The KWIC INDEX output presents all names used as input to the command and all possible variations in this name (via permutation about the dashes in each name) and lists these in alphabetical order. Since most names are selected for objects according to some criteria (if nothing more than all information pertaining to employees is preceded by the word "EMPLOYEE") this list may be referenced when assigning names to new objects to be added to the user requirement •

Implementation

One of the most common implementations of this command utilizes NAME-GEN:

```
NG  ALL

KWIC
```

This will give a KWIC INDEX of all names in the data base. From this, naming conventions can be devised (or imposed) for the user requirement . The results of this implementation is shown in Example 8.10a.

Options and Alternatives

The DIF parameter specifies the number of columns a name should be separated. It may be changed for ease in reading.

The FILE parameter is available, but the most effective way to use this command is through NAME-GEN.

Usage with NAME-GEN

Possibly a more practical form of this output is to generate a KWIC INDEX for each particular name type. Therefore, all PROCESSES names could be related, etc.

```
NG  PROCESS

KWIC
```

Example 8.10b presents this action.

```
        PSA VERSION 74032)                                           JUL  6, 1974  16:01.44
                                        ADS-EXAMPLE

                                      URA  KWIC  INDEX

PARAMETERS FOR: KWIC

   DIF=20

  SEQ      N A M E (PERMUTED)

  1 AND-EMPLOYEES                         DEPARTMENTS
  2 BLDG                     RM-228H-WEST-ENGINEERING
  3 CARD                                         TIME
  4 CHANGING-EMPLOYEE-DATA
  5 COMPANY-PROCEDURES-MANUAL
  6 DATA                            CHANGING-EMPLOYEE
  7 DEPARTMENT                                PAYROLL
  8 DEPARTMENTS-AND-EMPLOYEES
  9 EMP-PROCESSING                        TERMINATING
 10 EMPLOYEE
 11 EMPLOYEE-DATA                            CHANGING
 12 EMPLOYEE-INFO                         TERMINATING
 13 EMPLOYEE-INFORMATION                          NEW
 14 EMPLOYEE-INFORMATION                       WEEKLY
 15 EMPLOYEE-PROCESSING                           NEW
 16 EMPLOYEES                         DEPARTMENTS-AND
 17 ENGINEERING-BLDG                     RM-228H-WEST
 18 ERROR-LIST
 19 ERROR-LISTING-PRODUCTION
 20 HENRY-MILLER
 21 HIGHEST-LEVEL-PROCESS
 22 HIRED-TERMINATED-REPORT
 23 INFO                         TERMINATING-EMPLOYEE
 24 INFORMATION                          NEW-EMPLOYEE
 25 INFORMATION                        PAYROLL-MASTER
 26 INFORMATION                       WEEKLY-EMPLOYEE
 27 J-RATAJ                                    WALTER
 28 LEVEL-PROCESS                             HIGHEST
 29 LIST                                        ERROR
 30 LISTING-PRODUCTION                          ERROR
 31 L1
 32 MANUAL                         COMPANY-PROCEDURES
 33 MASTER-INFORMATION                        PAYROLL
 34 MILLER                                      HENRY
 35 NEW-EMPLOYEE-INFORMATION
```

Example 8.10a

```
URA VERSION 740328                                          JUL  6, 1974  16:01.44
                              ADS-EXAMPLE

                         U R A   K W I C   I N D E X

SEQ       N A M E (PERMUTED)

 36 NEW-EMPLOYEE-PROCESSING
 37 OUTPUTS                                              PAYSYSTEM
 38 PAY-STATEMENT
 39 PAYROLL-DEPARTMENT
 40 PAYROLL-MASTER-INFORMATION
 41 PAYROLL-PROCESSING
 42 PAYSTATEMENT-PRODUCTION
 43 PAYSYSTEM-OUTPUTS
 44 PROCEDURES-MANUAL                                      COMPANY
 45 PROCESS                                          HIGHEST-LEVEL
 46 PROCESSING                                             PAYROLL
 47 PROCESSING                                        NEW-EMPLOYEE
 48 PROCESSING                                     TERMINATING-EMP
 49 PRODUCTION                                        PAYSTATEMENT
 50 PRODUCTION                                       ERROR-LISTING
 51 RATAJ                                                 WALTER-J
 52 REPORT                                         HIRED-TERMINATED
 53 RM-228H-WEST-ENGINEERING-BLDG
 54 STATEMENT                                                  PAY
 55 SYSTEM                                                  TARGET
 56 TARGET-SYSTEM
 57 TERMINATED-REPORT                                        HIRED
 58 TERMINATING-EMP-PROCESSING
 59 TERMINATING-EMPLOYEE-INFO
 60 TIME-CARD
 61 WALTER-J-RATAJ
 62 WEEKLY-EMPLOYEE-INFORMATION
 63 WEST-ENGINEERING-BLDG                                   RM-228H
 64 228H-WEST-ENGINEERING-BLDG                                   RM
```

Example 8.10a (cont'd)

```
      URA VERSION 740710                                             JUL 28, 1974  21:34.54
                                              ADS-EXAMPLE

                                          U R A   K W I C   I N D E X

PARAMETERS FOR: KWIC

   DIF=20

  SEQ          N A M E (PERMUTED)

    1 CHECK-NEW                                    FIELD
    2 CHECK-PAYCALC                                FIELD
    3 CHECK-TERM                                   FIELD
    4 EMP-PRINTING                           TERMINATING
    5 EMP-PROCESSING                         TERMINATING
    6 EMP-UPDATING                           TERMINATING
    7 EMPLOYEE-PRINTING                              NEW
    8 EMPLOYEE-PROCESSING                            NEW
    9 EMPLOYEE-UPDATING                              NEW
   10 ERROR-LISTING-PRODUCTION
   11 FIELD-CHECK-NEW
   12 FIELD-CHECK-PAYCALC
   13 FIELD-CHECK-TERM
   14 FILE-REFERENCING
   15 INFO-VALIDATION                                NEW
   16 INFO-VALIDATION                               TERM
   17 INPUT-VALIDATION                           PAYCALC
   18 LISTING-PRODUCTION                           ERROR
   19 NEW                                    FIELD-CHECK
   20 NEW-EMPLOYEE-PRINTING
   21 NEW-EMPLOYEE-PROCESSING
   22 NEW-EMPLOYEE-UPDATING
   23 NEW-INFO-VALIDATION
   24 PAYCALC                                FIELD-CHECK
   25 PAYCALC-INPUT-VALIDATION
   26 PAYCALC-UPDATING
   27 PAYCHECK-PRINTING
   28 PAYROLL-PROCESSING
   29 PAYSTATEMENT-PRODUCTION
   30 PRINTING                                  PAYCHECK
   31 PRINTING                              NEW-EMPLOYEE
   32 PRINTING                           TERMINATING-EMP
   33 PROCESSING                                 PAYROLL
   34 PROCESSING                            NEW-EMPLOYEE
   35 PROCESSING                         TERMINATING-EMP
```

Example 8,10b

```
   URA  VERSION 740710                        (                              JUL 28, 1974   21:34.54
                                         ADS-EXAMPLE
                                    U R A   K W I C   I N D E X

SEQ          N A M E (PERMUTED)

 36 PRODUCTION                                   PAYSTATEMENT
 37 PRODUCTION                                  ERROR-LISTING
 38 REFERENCING                                          FILE
 39 TERM                                          FIELD-CHECK
 40 TERM-INFO-VALIDATION
 41 TERMINATING-EMP-PRINTING
 42 TERMINATING-EMP-PROCESSING
 43 TERMINATING-EMP-UPDATING
 44 UPDATING                                          PAYCALC
 45 UPDATING                                     NEW-EMPLOYEE
 46 UPDATING                                  TERMINATING-EMP
 47 VALIDATION                                       NEW-INFO
 48 VALIDATION                                      TERM-INFO
 49 VALIDATION                                  PAYCALC-INPUT
```

Example 8.10b(cont'd)

8.11 NAME-GEN (NG)

Report Description

The NAME GEN output is a list of names retrieved from the URA data base based on selection criteria specified by its parameters. Only one type of name may be retrieved for this report or all names in the data base can be presented.

Implementation

To get a list of all REAL-WORLD-ENTITY and PROCESS names:

```
NG  PROCESS  RWE
```

A NAME GEN output will be generated as well as a PUNCH file. Both will contain all the PROCESS and RWE names defined in the data base. With this list in the PUNCH file, it can be used as input to other URA commands. See Example 8.11a for the output generated by this command.

Options and Alternatives

There are many parameters available in specifying selection criteria. (See Part II , "URA COMMAND DESCRIPTIONS") Parameters are available for selecting particular name types (PROCESS, GROUP, SET, etc.) The ALL parameter can be used to specify all possible name types except SYNONYMS and UNDEFINED names:

```
NAME-GEN  ALL  (See Example 8.11b)
```

Within a given name type or group of names type you may select:

All names with a given KEYWORD=L1

```
NG  ALL  KEY=L1
```

All names (INPUT, OUTPUT or PROCESS) which are SUBPARTS to a given name:

```
NG  PROC  SO=PAYROLL-PROCESSING
```

The output for this is presented by Example 8.11c

All names with a given USER:

```
NG  ALL  PD=WALTER-J-RATAJ
```

If two or three of those parameters (KEY, SO, PD) are used:

```
NG  INPUT  SO=TIME-CARD  PD=WALTER-J-RATAJ  KEY=L-1
```

the only names that will appear in the NAME GEN output are those which satisfy all these selection criteria.

NAME-GEN utilizes a PUNCH file (defaults to URANAMES if one is not specified) so that the names generated by this command can be used as input to other report commands. For example, when

```
NG ALL
```

is given the names retrieved by this command are also put into the temporary file, URANAMES. You could then issue an FPS command for all these names by specifying:

```
FPS
```

This is equivalent to specifying the sequence:

```
NG ALL PUNCH= PUNCH

FPS F= PUNCH
```

When the goal of using the NAME-GEN command is to obtain PUNCH output, the NAME GEN report output may be omitted by giving the NOPRINT parameter:

```
NAME-GEN ALL NOPRINT
```

Usage with NAME-GEN

Not Applicable

Common Errors

Care should be taken that if the names retrieved are to be used as input to another command, that all the names are acceptable to the command. For example, you would not give:

```
NG GROUP PROCESS

CONTENTS
```

PROCESS names are not acceptable to the CONTENTS command (it only takes SET, INPUT, OUTPUT, ENTITY and GROUP names) and so errors would be encountered.

```
       URA  VERSION 740323                                         JUL   6, 1974  16:01.44
                                   ADS-EXAMPLE

                                          NAME GEN

PARAMETERS FOR: NG

   NOATTRIBUTE NOATTRIBUTE-VALUE NOCONDITION NOELEMENT NOENTITY NOEVENT NOGROUP NOINPUT
   NOINTERVAL NOKEYWORD NOMAILBOX NOMEMO NOOUTPUT NOUSER              PROCESS REAL-WORLD-ENTITY
   NORELATION NOSECURITY NOSET NOSOURCE NOSUBSETTING-CRITERION NOSYSTEM-PARAMETER NOUNDEFINED
   NOSYNONYMS BASIC PUNCH PRINT

   1     DEPARTMENTS-AND-EMPLOYEES              REAL-WORLD-ENTITY
   2     EMPLOYEE                               REAL-WORLD-ENTITY
   3     ERROR-LISTING-PRODUCTION               PROCESS
   4     NEW-EMPLOYEE-PROCESSING                PROCESS
   5     PAYROLL-DEPARTMENT                     REAL-WORLD-ENTITY
   6     PAYROLL-PROCESSING                     PROCESS
   7     PAYSTATEMENT-PRODUCTION                PROCESS
   8     TERMINATING-EMP-PROCESSING             PROCESS
```

Example 8.11a

```
      URA VERSION 740328                                          JUL  6, 1974  16:01.44
                                     ADS-EXAMPLE

                                            NAME GEN

PARAMETERS FOR: NG

   ATTRIBUTE ATTRIBUTE-VALUE CONDITION ELEMENT ENTITY EVENT GROUP INPUT INTERVAL KEYWORD MAILBOX
   MEMO OUTPUT USER                   PROCESS REAL-WORLD-ENTITY RELATION SECURITY SET SOURCE
   SUBSETTING-CRITERION SYSTEM-PARAMETER NOUNDEFINED NOSYNONYMS BASIC PUNCH PRINT

    1     CHANGING-EMPLOYEE-DATA                   ENTITY
    2     COMPANY-PROCEDURES-MANUAL                SOURCE
    3     DEPARTMENTS-AND-EMPLOYEES                REAL-WORLD-ENTITY
    4     EMPLOYEE                                 REAL-WORLD-ENTITY
    5     ERROR-LIST                               OUTPUT
    6     ERROR-LISTING-PRODUCTION                 PROCESS
    7     HENRY-MILLER                             USER
    8     HIGHEST-LEVEL-PROCESS                    KEYWORD
    9     HIRED-TERMINATED-REPORT                  OUTPUT
   10     L1                                       KEYWORD
   11     NEW-EMPLOYEE-INFORMATION                 INPUT
   12     NEW-EMPLOYEE-PROCESSING                  PROCESS
   13     PAY-STATEMENT                            OUTPUT
   14     PAYROLL-DEPARTMENT                       REAL-WORLD-ENTITY
   15     PAYROLL-MASTER-INFORMATION               SET
   16     PAYROLL-PROCESSING                       PROCESS
   17     PAYSTATEMENT-PRODUCTION                  PROCESS
   18     PAYSYSTEM-OUTPUTS                        OUTPUT
   19     RM-223H-WEST-ENGINEERING-BLDG            MAILBOX
   20     TARGET-SYSTEM                            KEYWORD
   21     TERMINATING-EMP-PROCESSING               PROCESS
   22     TERMINATING-EMPLOYEE-INFO                INPUT
   23     TIME-CARD                                INPUT
   24     WALTER-J-RATAJ                           USER
   25     WEEKLY-EMPLOYEE-INFORMATION              INPUT
```

Example 8.11b

```
    URA VERSION 740710                                                JUL 29, 1974  22:46.58
                                             ADS-EXAMPLE

                                              NAME GEN

PARAMETERS FOR: NG

    SUBPARTS-OF=PAYROLL-PROCESSING SUBLEVEL=0 NOATTRIBUTE NOATTRIBUTE-VALUE NOCONDITION NOELEMENT
    NOENTITY NOEVENT NOGROUP NOINPUT NOINTERVAL NOKEYWORD NOMAILBOX NOMEMO NOOUTPUT
    NOUSER             PROCESS NOREAL-WORLD-ENTITY NORELATION NOSECURITY NOSET NOSOURCE
    NOSUBSETTING-CRITERION NOSYSTEM-PARAMETER NOUNDEFINED NOSYNONYMS BASIC PUNCH PRINT

     1      ERROR-LISTING-PRODUCTION                     PROCESS
     2      FIELD-CHECK-NEW                              PROCESS
     3      FIELD-CHECK-PAYCALC                          PROCESS
     4      FIELD-CHECK-TERM                             PROCESS
     5      FILE-REFERENCING                             PROCESS
     6      NEW-EMPLOYEE-PRINTING                        PROCESS
     7      NEW-EMPLOYEE-PROCESSING                      PROCESS
     8      NEW-EMPLOYEE-UPDATING                        PROCESS
     9      NEW-INFO-VALIDATION                          PROCESS
    10      PAYCALC-INPUT-VALIDATION                     PROCESS
    11      PAYCALC-UPDATING                             PROCESS
    12      PAYCHECK-PRINTING                            PROCESS
    13      PAYSTATEMENT-PRODUCTION                      PROCESS
    14      TERM-INFO-VALIDATION                         PROCESS
    15      TERMINATING-EMP-PRINTING                     PROCESS
    16      TERMINATING-EMP-PROCESSING                   PROCESS
    17      TERMINATING-EMP-UPDATING                     PROCESS
```

Example . 8.11c

8.12 NAME-LIST (NL)

Report Description

The NAME LIST output is produced by this command. This list presents every name defined in the data base.

Implementation

All that is required to generate the output is:

```
NAME-LIST
```

The output shown in Example 8.12 will be produced.

Options and Alternatives

If it is desired for the names to be grouped by type (e.g., all PROCESSes together, all INPUTs together), the ORDER=BYTYPE parameter should be specified:

```
NAME-LIST   ORDER=BYTYPE
```

Usage with NAME-GEN

Not applicable

Common Errors

None

```
   RA  VERSION 740710                               OCT 23, 1974   10:47 02

                                         NAME LIST

        NAME                                TYPE                          SYNONYM

   74   birthdate                           GROUP                            -
   75   date                                GROUP                            -
   76   employee-name                       GROUP                            -
   77   employment-date                     GROUP                            -
   78   hourly-job-data                     GROUP                            -
   79   pay-date                            GROUP                            -
   80   personal-data                       GROUP                            -
   81   salaried-job-data                   GROUP                            -
   82   termination-date                    GROUP                            -
   83   employee-information                INPUT                         emp-info
   84   employment-termination-form         INPUT                         term info
   85   hourly-employment-form              INPUT                         h-emp-form
   86   hourly-employmet-form               INPUT                            -
   87   new-employee-information            INPUT                         new-info
   88   salaried-employment-form            INPUT                         s-emp-form
   89   tax-withholding-certificate         INPUT                         tax- ect
   90   tax-withholding-form                INPUT                            -
   91   time-card                           INPUT                         t-card
   92   month                               INTERVAL                         -
   93   week                                INTERVAL                         -
   94   year                                INTERVAL                         -
   95   check                               OUTPUT                           -
   96   error-listing                       OUTPUT                        e-list
   97   hire-report                         OUTPUT                           -
   98   hired-employee-report               OUTPUT                        hired-report
   99   hourly-employee-report              OUTPUT                        h-em -report
  100   pay-statement                       OUTPUT                        payc ebk
  101   pay-stub                            OUTPUT                           -
  102   paysysten-outouts                   OUTPUT                        payouts
  103   salaried-employee-report            OUTPUT                        s-emp-report
  104   terminated-employee-report          OUTPUT                        term-report
  105   error-documenting-procedure         PROCESS                          -
  106   hourly-employee-processing          PROCESS                          -
  107   new-employee-proce s$nc             PROCESS                          -
  108   paycheck-computation-procedure      PROCESS                          -
  109   payroll-processing                  PROCESS                       payp oc
  110   salaried-employee-processing        PROCESS                          -
  111   terminating-emp-processing          PROCESS                          -
  112   time-card-recovery-procedure        PROCESS                          -
```

Example 8.12

```
   JRA VERSION 740/10                                               OCT 23, 1974   10:47.02

                                            NAME LIST

       NAME                                 TYPE                                  SYNONYM

113    validity-check-procedure             PROCESS                                     -
114    departments-and-employees            REAL-WORLD-ENTITY                     dept emp
115    employee                             REAL-WORLD-ENTITY                     emp
116    payroll-department                   REAL-WORLD-ENTITY                     pay- dpt
117    dept-hourly-emp-relation             RELATION                                    -
118    dept-salaried-emp-relation           RELATION                                    -
119    payroll-master-information           SET                                   pay- mst
120    no-of-departments                    SYSTEM-PARAMETER                            -
121    no-of-hourly-employees               SYSTEM-PARAMETER                            -
122    no-of-payroll-processing             SYSTEM-PARAMETER                            -
123    no-of-salaried-employees             SYSTEM-PARAMETER                            -
124    one                                  SYSTEM-PARAMETER                            -
125    several                              SYSTEM-PARAMETER                            -
```

Example 8.12 (Continued)

8.13 PICTURE (PIC)

Report Description

This output (also called PICTURE) presents data in the URA data base in a graphical format. This output can be generated for any of the following name types:

```
SET
INPUT
OUTPUT
ENTITY
GROUP
ELEMENT
REAL-WORLD-ENTITY
PROCESS
```

The information retrieved is dependent on the type of name the report is being generated for. A separate PICTURE is given for each name used as input.

Implementation

To get a PICTURE for one name in the data base:

```
PICTURE  N=PAYCALC-UPDATING
```

The output generated for this is shown by Example 8.13a. All information relating this PROCESS to data (SETS, GROUPS, etc.), INPUTS, OUTPUTS and other PROCESSES will be presented by this report.

Options and Alternatives

To omit FLOW information from the PICTURE (information relating PROCESSES with INPUTS and OUTPUTS) specify "NOFLOW" as a parameter.

To omit DATA information from the PICTURE (information relating PROCESSES with SETS, ENTITIES, GROUPS, and ELEMENTS) specify "NODATA" as a parameter.

For example,

```
PICTURE  N=PAYROLL-PROCESSING  NODATA  NOFLOW
```

produces the output shown by Example 8.13b.

To omit STRUCTURE information from the PICTURE (information retrieval from the SUBPARTS or CONSISTS or SUBSETS statements for that name) specify "NOSTRUCTURE." Example 8.13c presents the output generated when given:

```
PICTURE  N=PAYROLL-PROCESSING  NOSTRUCTURE
```

The PICTURE output may be obtained for several names via the FILE parameter:

```
EDIT PIC.DATA NONUM NEW
PAY-STATEMENT
ERROR-LISTING
PAYSYSTEM-OUTPUTS
(blank line RETURN)
SE
EXEC CLI
     PIC FILE=PIC.DATA
```

An index to the output may be obtained by giving INDEX as a parameter.

Usage with NAME-GEN

NAME-GEN can be used to retrieve names for use by this command in the following manner:

```
NG  INPUT  OUTPUT
PIC
```

A PICTURE will be generated for each INPUT and OUTPUT name in the PSA data base. The most common use of this combination of commands is:

```
NG  PROCESS
PIC
```

Common Errors

For each name used as input to the command at least one page of output is generated. This can result in quite a lot to handle. It is best to send this output to *PRINT* via the SET command.

```
     URA VERSION 740710                                         JUL 29, 1974   22:46.58
                                          ADS-EXAMPLE
                                            PICTURE
PARAMETERS FOR: PIC

   NAME=PAYCALC-UPDATING NOINDEX DATA STRUCTURE FLOW
```

Example 8.13a

```
           URA VERSION 740710                               JUL 29, 1974  22:46.58
                                          ADS-EXAMPLE

                                        PROCESS PICTURE

         PAYCALC-UPDATING
                                           +--PROCESS--+
                                           IPAYSTATEME-I
                                           INT-PRODUCT-I
                                           IION        I
                                           +---PART----+
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
                                                 .
+---UNDEF---+                                    .                                       +---GROUP---+
IVALID-     I                                    .                                       I           I
IT-         I                                    .                                       I   CHECK   I
ICARD       I   .                          +--PROCESS--+                            .    I           I
+USES TO DRV+       .                      IPAYCALC-   I                        .        +--DERIVES--+
                           ..............IUPDATING   I..............
+---UNDEF---+       .                      I           I                        .        +---GROUP---+
IEMPLOYEE-  I   .                          +-----------+                            .    I           I
IDATA       I                                    .                                       I    STUB   I
I           I                                    .                                       I           I
+USES TO DRV+                                    .                                       +--DERIVES--+
                                                 .
                                                 .
                                                 .
                                               . . .
                                             .   .   .
                                           .     .     .
                                         .       .       .
                                       .         .         .
                                     .           .           .
                             +----SET----+  +--PROCESS--+  +--PROCESS--+
                             IPAYROLL-   I  IFIELD-     I  IFILE-      I
                             IMASTER-    I  ICHECK-     I  IREFERENCINGI
                             IINFORMATIONI  IPAYCALC    I  I           I
                             +--UPDATES--+  +-SUBPARTS--+  +-UTILIZES--+
```

Example 8,13a(cont'd)

```
      URA VERSION 740710                                            JUL 11, 1974   21:38.52
                                       ADS-EXAMPLE

                                         PICTURE
PARAMETERS FOR: PIC

   NAMF=PAYROLL-PROCESSING NUINDEX NODATA STRUCTURE NOFLOW
```

Example 8.13b

```
URA VERSION 740710                                     JUL 11, 1974  21:38.52
                                  ADS-EXAMPLE
                             PROCESS PICTURE
PAYROLL-PROCESSING





                               +--PROCESS--+
                               IPAYROLL-   I
                               IPROCESSING I
                               I           I
                               +-----------+
                                     .
                                     .
                                     .
                                    ...
                                 .   .   .
                              .      .      .
                           .       . . .       .
                        .        .   .   .        .
                     .         .     .     .         .
                  .          .       .       .          .
               .           .         .         .           .
            .            .           .           .            .
   +--PROCESS--+ +--PROCESS--+ +--PROCESS--+ +--PROCESS--+ +--PROCESS--+
   IFILE-      I INEW-       I ITERMINATIN-I IPAYSTATEME-I IERROR-     I
   IREFERENCINGI IEMPLOYEE-  I IG-EMP-PROC-I INT-PRODUCT-I ILISTING-   I
   I           I IPROCESSING I IESSING     I IION        I IPRODUCTION I
   +-SUBPARTS--+ +-SUBPARTS--+ +-SUBPARTS--+ +-SUBPARTS--+ +-SUBPARTS--+
```

Example B.13b(cont'd)

```
      URA  VERSION 740710                                                 JUL 11, 1974   21:38.52
                                        ADS-EXAMPLE
                                          PICTURE
PARAMETERS FOR: PIC

   NAME=PAYROLL-PROCESSING NOINDEX DATA NOSTRUCTURE FLOW
```

Example 8.13c

```
          JRA VERSION 740710                                     JUL 11, 1974  21:38.52
                                             ADS-EXAMPLE

                                        PROCESS PICTURE

         PAYROLL-PROCESSING




+---INPUT---+                                                                         +--OUTPUT---+
I           I                                                                         IPAY-       I
I TIME-CARD I                                                                         ISTATEMENT  I
I           I  .                                                                   .  I           I
+USES TO DRV+      .                                                            .     +--DERIVES--+
                      .                                                      .
+---INPUT---+            .                                                .           +--OUTPUT---+
ITERMINATIN-I               .                                          .              IERROR-     I
IG-EMPLOYEE-I                  .                                    .                 ILISTING    I
I-INFO      I    .                .        +--PROCESS--+         .               .    I           I
+USES TO DRV+          .             .     IPAYROLL-   I      .             .         +--DERIVES--+
                            ...............IPROCESSING I................
+---INPUT---+          .             .     I           I      .             .         +--OUTPUT---+
INEW-       I    .                .        +-----------+         .               .    IHIRED-     I
IEMPLOYEE-  I                  .                 .                  .                 ITERMINATED-I
IINFORMATIONI               .                    .                     .              IREPORT     I
+USES TO DRV+            .                       .                        .           +--DERIVES--+
                      .                          .                           .
+---INPUT---+      .                             .                              .     +--OUTPUT---+
IWEEKLY-    I  .                                 .                                 .  IPAYSYSTEM- I
IEMPLOYEE-  I                                   . .                                   IOUTPUTS    I
IINFORMATIONI                                  .   .                                  I           I
+-RECEIVES--+                                 .     .                                 +-GENERATES-+
                                             .       .
                                            .         .
                                           .           .
                                    +--ENTITY---+ +----SET----+
                                    IVARYING-   I IPAYROLL-   I
                                    IEMPLOYEE-  I IMASTER-    I
                                    IDATA       I IINFORMATIONI
                                    +--UPDATES--+ +--UPDATES--+
```

Example 8.13c (cont'd)

8.14 PRINT ATTRIBUTE-VALUES

Report Description

The report generated by this command is called the ATTRIBUTE REPORT and presents, for each ATTRIBUTE name used as input to the command, all those names in the data base it is an ATTRIBUTE for and the value it takes on for that name.

Implementation

To get the basic report:

```
PAV  N=TYPE
```

The output generated is shown in Example 8.14.

Options and Alternatives

You may also enter ATTRIBUTE names into a file to be used as input to the command:

```
EDIT PAV.DATA NONUM NEW
TYPE
MODE
(blank line RETURN)
SE
EXEC CLI
     PAV FILE=PAV.DATA
```

Usage with NAME-GEN

This command can effectively be used in conjunction with NAME-GEN in the following manner:

```
NG ATTRIBUTE
PAV
```

Information will be presented for all ATTRIBUTES defined in the data base.

Common Errors

It is necessary that only ATTRIBUTE names be used as input to the command. Any other name types will generate error diagnostics.

```
      URA VERSION 740710                                   JUL 29, 1974  22:46.58
                                  ADS-EXAMPLE

                                ATTRIBUTE REPORT

PARAMETERS FOR: PAV

  NAME=TYPE


   1*  ATTRIBUTE: TYPE


          APPLIES TO:                             VALUE:
         1 OCCURRENCE-OF-BAD-INPUT                RANDOM-EVENT
         2 NUMBER-OF-DEPENDENTS                   NUMERIC-INFORMATION
         3 PAYRATE                                NUMERIC-INFORMATION
         4 EMPLOYEE-NAME                          CHARACTER-INFORMATION
         5 COMPLETE-PAY-INFORMATION               MAINTAINED-WEEKLY
         6 TAX-RATE                               NUMERIC-INFORMATION
         7 YTD-DEDUCT                             NUMERIC-INFORMATION
         8 YTD-GROSS                              NUMERIC-INFORMATION
         9 GROSS-PAY                              NUMERIC-INFORMATION
        10 NET-PAY                                NUMERIC-INFORMATION
        11 TERMINATION-CODE                       NUMERIC-INFORMATION
        12 DEDUCTIONS                             NUMERIC-INFORMATION
        13 PAYSTATEMENT-PRODUCTION                WEEKLY-PROCESS
        14 TERMINATING-EMPLOYEE-INFO              RECURRING-WEEKLY-INPUT
        15 NEW-EMPLOYEE-INFORMATION               RECURRING-WEEKLY-INPUT
        16 TIME-CARD                              RECURRING-WEEKLY-INPUT
        17 TERMINATING-EMP-PROCESSING             RANDOM-PROCESS
        18 NEW-EMPLOYEE-PROCESSING                RANDOM-PROCESS
        19 PAYROLL-MASTER-INFORMATION             IMS-FORMAT
        20 ERROR-LISTING-PRODUCTION               RANDOM-PROCESS
        21 HIRED-TERMINATED-REPORT                RANDOM-OUTPUT
        22 ERROR-LISTING                          RANDOM-OUTPUT
        23 PAY-STATEMENT                          RECURRING-OUTPUT
```

Example 8.14

8.15 PROCESS-INPUT-OUTPUT (PRIO)

Report Description

This command produces the PROCESS INPUT/OUTPUT report. For those PROCESS names used as input to the command it retrieves data flow information (via the USES, RECEIVES, GENERATES, DERIVES and UPDATES relationships specified in the data base) and presents it in a narrative, outline format.

Implementation

To generate the basic form of the report for one name:

```
PRIO   N=PAYROLL-PROCESSING
```

This generates the information in the format shown by Example 8.15a.

Options and Alternatives

Any of the information in the report can be omitted by specifying various parameters. For example, NDESC can be given and the DESCRIPTION comment entry associated with the PROCESS names will not be printed.

The PROCEDURE comment entry associated with PROCESS names can be included in the report when the "PRCD" parameter is also given.

The format of the report can be altered somewhat; each PROCESS can have the information about it printed on a separate page when the "NEW-PAGE" parameter is given.

Of course this report can be generated for several names when the "FILE" parameter is specified.

Below is an example illustrating some of the options described above:

```
EDIT PRIO.DATA NONUM NEW BLOCK(800)
NEW-EMPLOYEE-PRINTING
NEW-EMPLOYEE-UPDATING
(blank line RETURN)
SE
EXEC CLI
      PRIO F=PRIO.DATA NDESC NPG PRCD
```

This procedure produces the report for the two names in the file 'T.DATA', the PROCEDURE comment entry is printed instead of the DESCRIPTION comment entry, and each PROCESS is described on a new page. Example 8.15b presents the outcome of this procedure.

Another option available is to utilize the PUNCH facility to obtain a list of data names (SETS, INPUTS, OUTPUTS, ENTITIES, GROUP and ELEMENTS) related to the PROCESSES in the report. This may be desirable to be used as input to another report command:

```
PRIO  N=PAYROLL-PROCESSING  NP PUNCH= PUNCH
DATA PROCESS  F= PUNCH  DATA
```

In this implementation, the data associated with PAYROLL-PROCESSING is being used as input to the DATA-PROCESS command to see how it realtes to other PROCESSES.

Usage with NAME-GEN

It is quite easy to use NAME-GEN to retrieve PROCESS names to be used as input to the command:

```
NG  PROCESS
PRIO
```

The report will be generated for all PROCESS names in the data base.

Common Errors

The main thing to watch out for is that all names used as input are PROCESS names.

```
    URA  VERSION 740710                                          AUG 12, 1974   20:54.50
                                           ADS-EXAMPLE
                                       PROCESS INPUT/OUTPUT

PARAMETERS FOR: PRIO

 NAME=PAYPROC INPUTS OUTPUTS DESCRIPTION NOPROCEDURE PRINT NOPUNCH


  1*  PAYROLL-PROCESSING

          THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
          IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
          ALL INPUTS AND PRODUCES ALL OUTPUTS.

                        ***  INPUTS  ***

           1 WEEKLY-EMPLOYEE-INFORMATION          RECEIVED
           2 TIME-CARD                            USED TO DERIVE

                        ***  OUTPUTS  ***

           1 PAYSYSTEM-OUTPUTS                    GENERATED
           2 PAY-STATEMENT                        DERIVED
           3 ERROR-LISTING                        DERIVED
           4 HIRED-TERMINATED-REPORT              DERIVED
           5 VARYING-EMPLOYEE-DATA                UPDATED
           6 FIXED-EMPLOYEE-DATA                  UPDATED
           7 PAYROLL-MASTER-INFORMATION           UPDATED
```

Example 8.15a

```
  ,RA  VEFSION 74071U                                   AUG 12, 1974   20:54.50
                            ADS-EXAMPLE
                                 PROCESS INPUT/OUTPUT
PARAMETERS FOR: PRIO
 FILE INPUTS OUTPUTS NODESCRIPTION PROCEDURE PRINT NOPUNCH
```

Example 8.15b

```
URA VERSION 740710                                          AUG 12, 1974  20:54.50
                                  ADS-EXAMPLE

                              PROCESS INPUT/OUTPUT

1*  NEW-EMPLOYEE-PRINTING

    1-) ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
    2-) SAVE THE INFORMATION
    3-) PRINT THE NEW HIRE SECTION OF THE HT REPORT.

                       ***  INPUTS  ***

     1 VALID-NEW-INFO                        USED TO DERIVE

                       ***  OUTPUTS  ***

     1 NEW-EMPLOYEE-PART                     DERIVED
```

Example 8.15b (cont'd)

```
URA VEPSION 740710                                           AUG 12, 1974  20:54.50
                                        ADS-EXAMPLE

                                  PROCESS INPUT/OUTPUT

2*  NEW-EMPLOYEE-UPDATING

        1-) OBTAIN A UNIT OF VALID NEW INFORMATICN
        2-) INSERT THIS INFORMATION INTO RESPECTIVE AREAS
            OF THE DATA BASE
        3-) INSERT ZEROS IN THOSE ELEMENTS NOT IN THE
            VALID NEW INFO.

                        ***  INPUTS  ***

         1 VARYING-EMPLOYEE-DATA                USED
         2 FIXED-EMPLOYEE-DATA                  USED
         3 VALID-NEW-INFO                       USED TO DERIVE

                        ***  OUTPUTS  ***

         1 EMPLOYEE-DATA                        DERIVED
         2 PAYROLL-MASTER-INFORMATION           UPDATED
```

Example 8.15b (cont'd)

8.16 PUNCH-COMMENT-ENTRY (PCOM)

Report Description

The PUNCHED COMMENT ENTRIES output is generated by this command. It presents selected comment entries for each name used as input to the command. Any type of name may be used as input to the command. Depending on the type of name the following comment entries may be retrieved:

DERIVATION	(DER)
DESCRIPTION	(DESC)
FALSE-WHILE	(FW)
PROCEDURE	(PRCD)
TRUE-WHILE	(TW)
VOLATILITY	(VOL)
VOLATILITY-MEMBER	(VOLM)
VOLATILITY-SET	(VOLS)

Implementation

To obtain the DESCRIPTION comment entry for one name:

```
PCOM  N=PAYROLL-PROCESSING  DESC
```

This will generate the report shown in Example 8.16a. A PUNCH file will also be generated with the same information as the report. This file can then be editted and used as input to the RCOM command:

```
EDIT URAPCOM
(edit commands)
SE
EXEC CLI
RCOM I=URAPCOM
```

Options and Alternatives

The FILE parameter allows comment entries generated for several names:

```
EDIT PCOM.DATA NONUM NEW
EMPLOYEE
TIME-CARD
PAY-STATEMENT
(blank line RETURN)
SE
EXEC CLI
     PCOM F=PCOM.DATA DESC
```

The output for this is given in Example 8.16b.

Multiple comment entries can also be generated for several names:

```
EDIT PCM.DATA NONUM NEW
NEW-EMPLOYEE-PRINTING
NEW-INFO-VALIDATION
(blank line RETURN)
SE
EXEC CLI
     PCOM F=PCM.DATA DESC PRCD
```

The resulting output is shown in Example 8.16c.

When the objective of implementing this command is to generate a PUNCH file, printing of the report may be surpressed by issuing NOPRINT so the command may appear as:

```
PCOM  F=T.DATA PUNCH= PUNCH  NOPRINT
```

Usage with NAME-GEN

One or more comment entries may be retrieved for a given name type:

```
NG  RWE
PCOM  DESC
```

or for several name types:

```
NG  SET  PROCESS
PCOM  DESC  PRCD
```

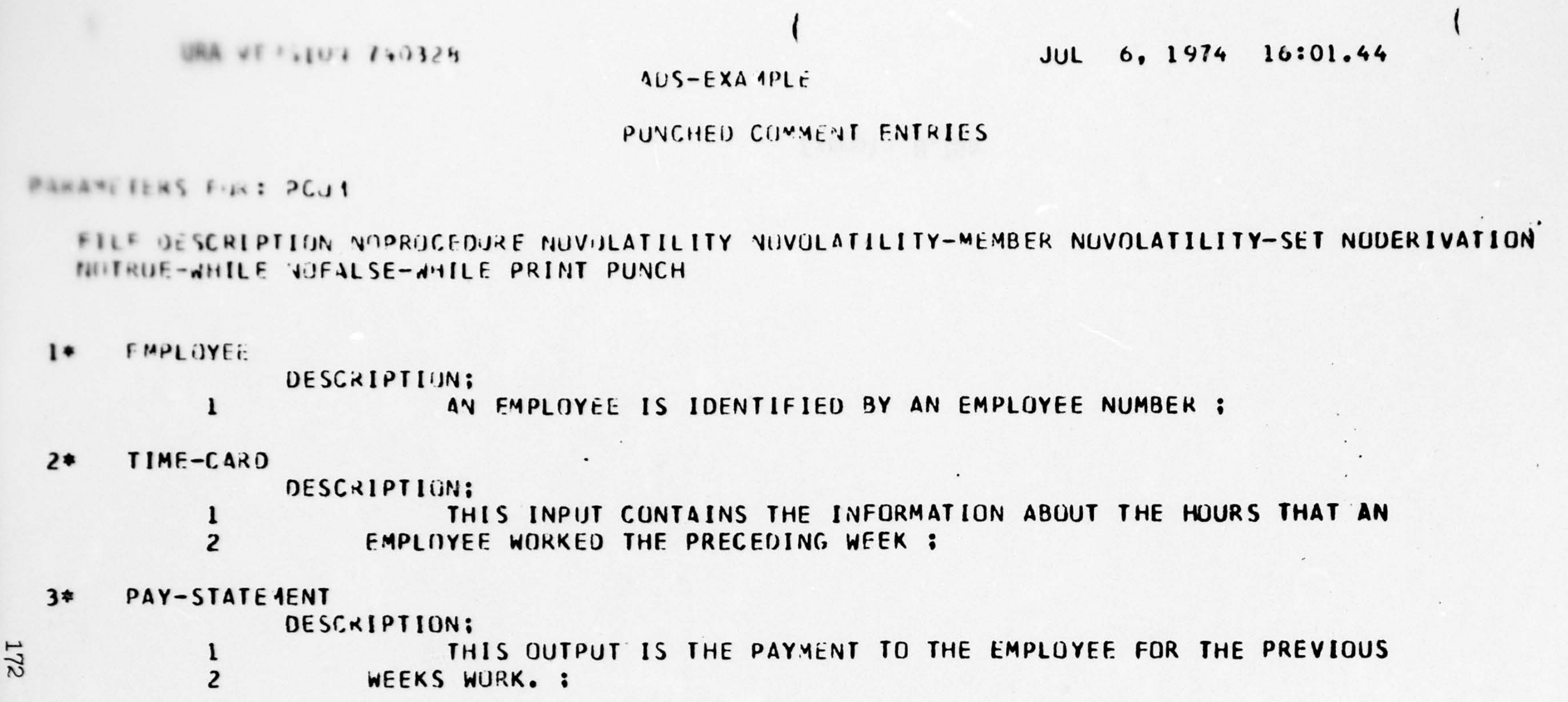
Notice that the PRCD parameter is given, but SETS cannot have PROCEDURE statements associated with them. Only the DESCRIPTION statements will be retrieved for SET names while both DESCRIPTION and PROCEDURE statements will be retrieved for PROCESS names.

Common Errors

The user should note that most of the parameters (FALSE-WHILE, VOLATILITY, etc.) can only apply to one type of name (CONDITION, ENTITY, respectively).

```
     URA VERSION 740710                                              JUL 29, 1974  22:46.58
                                      ADS-EXAMPLE

                                 PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PC01

   NAME=PAYROLL-PROCESSING DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER
   NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH


 1*   PAYROLL-PROCESSING
              DESCRIPTION;
        1               THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
        2               IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
        3               ALL INPUTS AND PRODUCES ALL OUTPUTS. ;
```

Example 8.16a



```
URA VERSION 740325                                  JUL  6, 1974  16:01.44
                              ADS-EXAMPLE

                        PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PC01

   FILE DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH


1*    EMPLOYEE
               DESCRIPTION;
        1              AN EMPLOYEE IS IDENTIFIED BY AN EMPLOYEE NUMBER ;

2*    TIME-CARD
               DESCRIPTION;
        1              THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN
        2          EMPLOYEE WORKED THE PRECEDING WEEK ;

3*    PAY-STATEMENT
               DESCRIPTION;
        1              THIS OUTPUT IS THE PAYMENT TO THE EMPLOYEE FOR THE PREVIOUS
        2          WEEKS WORK. ;
```

Example-8.16b

```
URA VERSION 740328                                      JUN  8, 1974  14:04.08

                              PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PCOM

  FILE DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
  NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

 1*   NEW-EMPLOYEE-PRINTING
            DESCRIPTION;
        1               THIS PROCESS PRODUCES THE NEW HIRE SECTION OF THE H-T REPORT. ;

 2*   NEW-EMPLOYEE-PRINTING
            PROCEDURE;
        1               1-) ACCEPT A VALID UNIT OF NEW EMPLOYEE INFORMATION
        2               2-) SAVE THE INFORMATION
        3               3-) PRINT THE NEW HIRE SECTION OF THE HT REPORT. ;

 3*   NEW-INFO-VALIDATION
            DESCRIPTION;
        1               THIS PROCESS ACCEPTS CORRECT INPUT INFORMATION AND
        2               REJECTS THE INPUT OTHERWISE. ;

 4*   NEW-INFO-VALIDATION
            PROCEDURE;
        1               1-) READ A UNIT OF NEW EMPLOYEE INFORMATION
        2               2-) CHECK THE RANGES OF THE FIELDS
        3               3-) IF: FIELDS CORRECT
        4                      THEN: ADD TO THE DATA BASE
        5                      ELSE: REJECT ENTIRE UNIT OF INFORMATION ;
```

Example - 8.16c

8.17 STRUCTURE (STR)

Report Description

The STRUCTURE output is generated by this command. It presents structures in the user requirement specified through usage of the SUBPARTS statement. For this reason, the STRUCTURE output is only available for REAL-WORLD-ENTITIES, INPUTS, OUTPUTS and PROCESSES defined in the user requirement.

Implementation

To get the STRUCTURE report for all PROCESSES defined in the data base specify:

```
STRUCTURE
```

The resulting output is shown by Example 8.17a.

Options and Alternatives

The STRUCTURE report may be obtained for either INPUTS or OUTPUTS or REAL-WORLD-ENTITIES or PROCESSES. To generate a report for INPUTS:

```
STR INPUT
```

This output is shown by Example 8.17b.

The format of the output can be altered somewhat by reassigning a value for INDENT. It specifies the number of spaces to indent each succeeding level of the structure. We could do this by:

```
STR INDENT=10
```

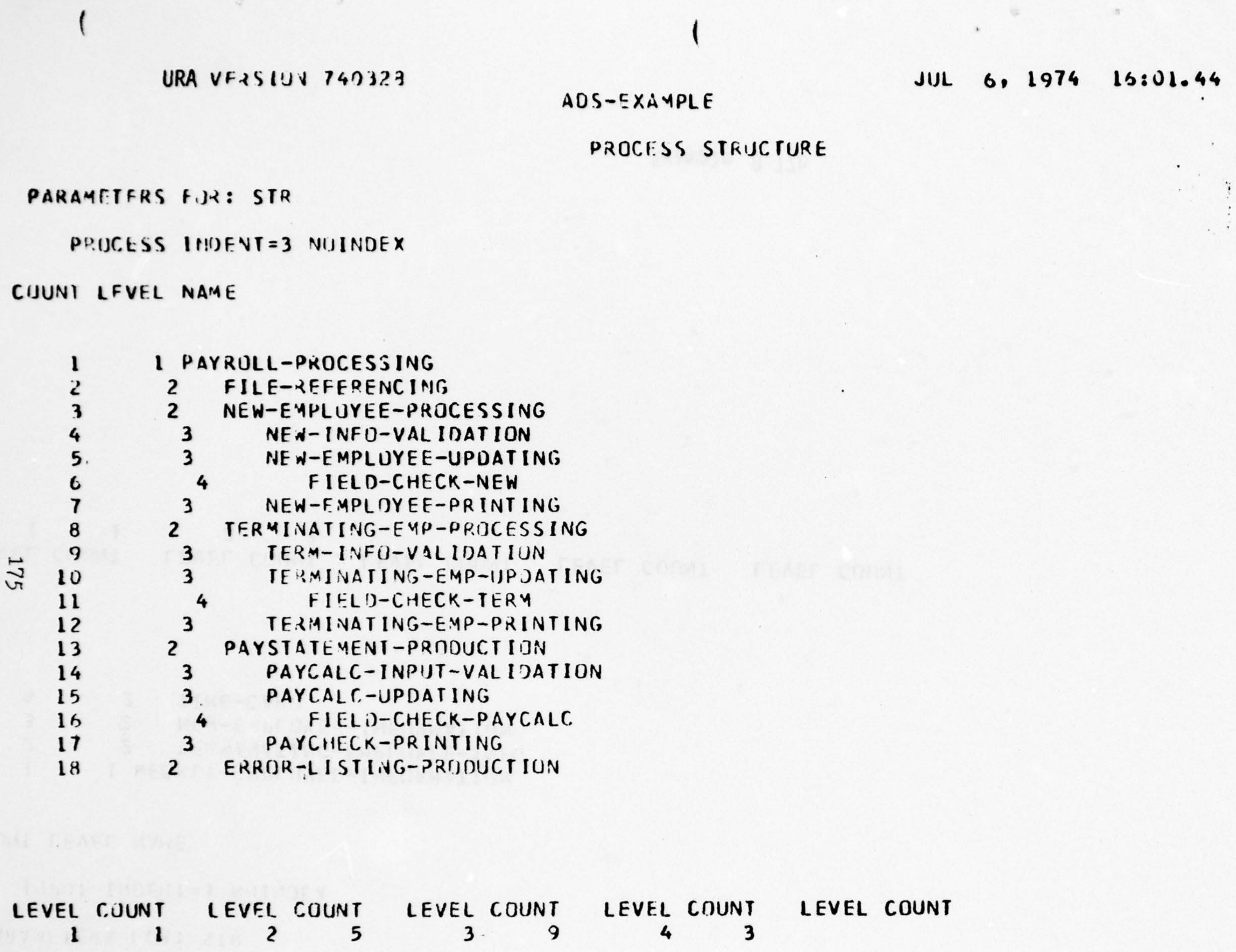
INDENT can take on any integer value, 1 through 10.

Usage with NAME-GEN

Not applicable.

Common Errors

None.



```
     URA VERSION 740323                                    JUL  6, 1974  16:01.44
                               ADS-EXAMPLE

                                  PROCESS STRUCTURE

 PARAMETERS FOR: STR

    PROCESS INDENT=3 NOINDEX

COUNT LEVEL NAME

     1      1 PAYROLL-PROCESSING
     2       2   FILE-REFERENCING
     3       2   NEW-EMPLOYEE-PROCESSING
     4        3     NEW-INFO-VALIDATION
     5        3     NEW-EMPLOYEE-UPDATING
     6         4       FIELD-CHECK-NEW
     7        3     NEW-EMPLOYEE-PRINTING
     8       2   TERMINATING-EMP-PROCESSING
     9        3     TERM-INFO-VALIDATION
    10        3     TERMINATING-EMP-UPDATING
    11         4       FIELD-CHECK-TERM
    12        3     TERMINATING-EMP-PRINTING
    13       2   PAYSTATEMENT-PRODUCTION
    14        3     PAYCALC-INPUT-VALIDATION
    15        3     PAYCALC-UPDATING
    16         4       FIELD-CHECK-PAYCALC
    17        3     PAYCHECK-PRINTING
    18       2   ERROR-LISTING-PRODUCTION


LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT   LEVEL COUNT
  1     1       2     5       3     9       4     3
```

Example 8.17a

```
     URA VERSION 740710                                          JUL 29, 1974  22:46.58
                                     ADS-EXAMPLE

                                   INPUT STRUCTURE

PARAMETERS FOR: STR
  INPUT INDENT=3 NOINDEX

COUNT LEVEL NAME

   1     1 WEEKLY-EMPLOYEE-INFORMATION
   2      2   TERMINATING-EMPLOYEE-INFO
   3      2   NEW-EMPLOYEE-INFORMATION
   4      2   TIME-CARD

LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT
   1     1        2     3
```

Example 8.17b

8.18 SUMMAF((SUM)

Report Description

The DATA BASE SUMMARY is produced by this command. This report presents statistical information concerning the use of each name type possible in the URA data base. For example, this report presents the number of PROCESSES, SETS, etc. that have been defined in the data base as well as the number of these which have DESCRIPTION and SYNONYM relationships assigned to them.

Implementation

There is only one way to implement this report:

```
SUM
```

The DATA BASE SUMMARY for a small data base was produced as shown in Example 8.18a.

Options and Alternatives

None.

Usage with NAME-GEN

Not applicable.

Common Errors

None.

```
URA VERSION 740328                                           JUL  8, 1974  16:01.44
                                   ADS-EXAMPL

                                 DATA BASE SUMMARY

                                 COUNT #W/SYN PERCENT #W/DESC PERCENT
*** UNKNOWN OR AMBIGUOUS ***       5      2    40.00      0
ATTRIBUTE                          1      0               0
ATTRIBUTE-VALUE                   10      3    30.00      0
CONDITION                          1      0               0
ELEMENT                           13      8    61.53     10    76.92
ENTITY                             2      2   100.00      2   100.00
EVENT                              5      1    20.00      1    20.00
GROUP                              8      5    62.50      8   100.00
INPUT                              4      4   100.00      4   100.00
INTERVAL                           2      1    50.00      2   100.00
KEYWORD                            3      0               0
MAILBOX                            1      0               0
MEMO                               1      0               1   100.00
OUTPUT                             4      4   100.00      4   100.00
USER                               2      2   100.00      2   100.00
PROCESS                           18     13    72.22     12    66.66
REAL-WORLD-ENTITY                  3      3   100.00      3   100.00
RELATION                           1      1   100.00      1   100.00
SECURITY                           2      0               0
SET                                3      1    33.33      1    33.33
SOURCE                             1      0               0
SYSTEM-PARAMETER                   3      0               0

 ** TOTAL **                      93     50    53.76     51    54.83
```

Example 8.18a

9. Error Diagnostics

URA has extensive error checking facilities to aid the users in preventing errors in their problem statement. At the URA command mode level, checks are made that all commands given are legal URA commands and all parameters given are legal parameters for that command. If an illegal command is given, an illegal parameter, or illegal parameter for that command, the following message will be generated:

```
INVALID PARAMETER -
ENTER REPLACEMENT OR BLANK LINE
?
```

The user must enter the replacement following the question mark and then hit the carriage return key. If the command is accepted, processing of that command commences. Should an error be encountered while processing the command, one of the following three types of error diagnostics will be given:

i) Data Base Management System Errors

These errors are encountered when there may be some danger of destroying the contents of the URA data base or there is a bug in the URA software. Even though the URA software might be the cause of the error, it is doubtful if it will do anything to harm the contents of the users data base.

```
*** ERROR 16 - DATA BASE FILE INCONSISTENT
```

This error message is given if the user attempts to modify or retrieve information from a URA data base which has had its contents altered so that it is unusable by the URA software. There is no way to recover from this type of error unless a save file is kept (See the SAVE command in Section 8.2).

```
*** ATTENTION TRAPPED BY D.B. HANDLER
```

This message is given whenever the user uses the Break button (control-E) to get out of URA mode. To return to URA the user must issue the EXEC CLI command. If an ATTENTION is given by the user while URA is processing a modifier command (like INPUT-URL), it is necessary that the user issue EXEC CLI or the contents of the URA data may become unusable. There is no such danger if URA is processing a report command.

**** ERROR # n FOUND IN ROUTINE # m

An error message of this format usually designates a bug in the URA software, where n is the error number. If the values of the variables n and m are 16 and 30 respectively, the error designates a data base inconsistency which is usually a user error. Any other errors of this form with different values should be brought to the attention of those persons maintaining the URA software.

ii) URA Command Errors

These errors are encountered in the processing of URA commands and are user errors. These diagnostics are generated when the user presents ambiguous or incorrect information to the commands. In most cases, URA will take no action to fulfill the users request if an error is encountered. The command must be restated corrected form, before action is taken. All these commands are presented in the following format:

URAnnn:subroutine: error-message

where "nnn" designates the URA error number, "subroutine", the subroutine in the URA software where the error occurred, and "error-message" some diagnostics to why an error was encountered.

iii) URA Input Errors

These errors are a specific type of URA command error which are encountered when using the INPUT-URL or DELETE-URL commands incorrectly. URA always attempts to recover from these errors unless an excess number have been encountered. Each of these errors are assigned a level number, 1 through 4. The user is allowed to make up to 24 level 1 and 24 level 2 errors, but a single level 3 or level 4 error will terminate processing of the command. The levels are described below.

Level	Description	Limit
1	Warning	24
2	Serious user error	24
3	URA unable to recover	0
4	Exceeded URA capabilities	0

These types of errors are presented in the following format:

```
**** LEVEL j, URAnnn:subroutine:error-message
```

where "j" designates the level number and "nnn", the URA error number. The last part of the format is the same as the standard URA command errors.

After processing any URA command, a STOP status message is given. This message designates that processing of the command was successful (STOP 0, i.e., errors were handled effectively etc.), or that processing was not totally successful (STOP 4). STOP 4 is given when a level limit is exceeded for INPUT-URL errors, for example. The following is a list of all possible errors that can be encountered while using URA. A short description of each command accompanies it as well as suggested action to take should the error occur.

Number	Subroutine	Error Message
2	NLEX:	NAME TOO LONG A user defined name has exceeded the 30 character limit allowed by URL/URA. The name is truncated to 30, but is still put in data base. See Section 10.1 of this paper.
3	NLEX:	'EOF' NOT FOUND BEFORE END-OF-FILE The user has terminated the input by a $ENDFILE (or CONTROL-C) rather than the URL 'EOF'. Processing of the input is terminated.
4	NLEX:	ILLEGAL CHARACTER - TREATED AS BLANK An illegal character encountered in an input line or legal URL character is used in the wrong context. See ESD TR # 75-88, Vol II for list of legal characters. Statement with this error may not be deleted.
5	NLEX:	END-OF-FILE IN MIDDLE OF COMMENT A $ENDFILE (or CONTROL-C) has been encountered following the '/*' comment characters. Processing of the input is terminated.
6	SCAN:	INVALID LEXICAL TYPE RETURNED FROM NLEX URA software error. Please notify persons maintaining URA should this error occur.
7	SCAN:	ILLEGAL CHARACTER - IGNORED An illegal character encountered when scanning an input line. See ESD TR # 75-88, Vol II for complete list of legal characters. Statement with this error may not be deleted.
8	COMLOP:	PARSE STACK OVERFLOW URA software error. Please notify persons maintaining URA should this error occur.
9	PROK:	BAD CASE URA software error. Please notify persons maintaining URA should this error occur.
10	REDUCE:	NO APPLICABLE PRODUCTION - SYNTAX ERROR - START SKIPPING Illegal URL statement syntax is encountered. If this is a header statement, following statements will be assigned to the previous header statement. Error may be result of incorrect usage of a URL reserved word, see Section 10.1.
11	STACK:	ILLEGAL SYMBOL PAIR - SYNTAX ERROR - START SKIPPING Illegal URL statement syntax is encountered. If this is a header statement, following statements will be assigned to the previous header statement. This statement is not entered into the URA data base. (See Section 10.1).

Number	Subroutine	Error Message
12	SYMBOL:	SYMBOL TABLE OVERFLOW Exceeded limits of URA. Reissue INPUT-URL command at point in the input file where this error occurred.
13	SYMBOL:	TOO MANY SYMBOLS Exceeded limits of URA. Reissue INPUT-URL command at point in the input file where this error occurred.
14	SETYPE:	INVALID SYMBOL TABLE POINTER URA software error. Please notify persons maintaining URA should this error occur.
15	STACK:	INVALID CASE URA software error. Please notify persons maintaining URA should this error occur.
16	COMENT:	END-OF-FILE IN COMMENT ENTRY A $ENDFILE (or CONTROL-C) encountered in URL comment entry. Processing of the input is terminated.
17	SKIP:	END OF FILE WHILE SKIPPING Serious error. In attempt to recover from previous errors the end of input has been encountered. Processing of input is terminated.
18	IDENT:	NO NAMES IN DATA BASE Attempt to retrieve names from an empty data base.
19	RECOV:	UNABLE TO RECOVER AT THIS TIME Processing of input is terminated due to serious errors which make it unable to continue.
20	RECOV:	LAST STATEMENT SKIPPED Statement where error occurred is skipped so that processing of input may continue.
21	SETINF:	INVALID SYMBOL TABLE POINTER URA software error. Please notify persons maintaining URA should this error occur.
22	OTHERS:	SAME ATTRIBUTE ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE An attempt was made to assign a second value to the same ATTRIBUTE for a given name. The new value is ignored.
23	SUBPRT:	TOO MANY LEVELS - STACK OVERFLOW The limit allowed for retrieving names via the SUBPARTS-OF parameter has been exceeded. Reissue the NAME-GEN command with the last name retrieved as the value for the SUBPARTS-OF parameter.

Number	Subroutine	Error Message
24	MAINRCOM:	MISSING SEMICOLON ON LINE AFTER NAME Semicolon is needed to terminate comment entry statement.
25	HEAD:	INVALID HEADER STATEMENT - STATEMENTS WILL BE IGNORED Illegal syntax of header statement. All URL statements up to the next header statement will be ignored.
26	IGINFO:	INVALID SYMBOL TABLE POINTER URA software error. Please notify persons maintaining URA should this error occur.
27	PTABIN:	INVALID LEXICAL TYPE OR END-OF-FILE URA software error. Please notify persons maintaining URA should this error occur.
29	CHKREL:	CONFLICT WITH EXISTING CONNECTIONS (RELB) RELTYP # Attempt made to change name type to one which conflicts with the context in which the name is used. No change is made.
30	CHKREL:	CONFLICT WITH EXISTING CONNECTIONS (RELB) RELTYP # Attempt made to change name type to one which conflicts with the context in which the name is used. No change is made.
31	MAINCT:	BAD INPUT FORMAT The format of the file used as in put to the command is incorrect. See the command description for correct format. No change is made.
32	MAINCT:	NAME NOT IN DATA BASE Attempt to change name tupe of name not defined in the URA data base.
33	MAINCT:	INVALID NAME TYPE Attempt to assign an illegal name type to a name. Probably a spelling error.
34	MAINCT:	NAME TYPE TOO LONG Attempt to assign an illegal name type to a name. Probably a spelling error.
35	MAINCT:	WARNING - STUFF AFTER NAME TYPE The input file contains more than just name and new name type. The extra data will be ignored by the command.
36	MAINCT:	INVALID NAME - TOO LONG The name for which the change is to be made is over 30 characters. Check spelling.

Number	Subrout ne	Error Message
37	FNDPD:	THIS IS NOT A PD FOR ANY NAMES - This problem definer is not associated to any names defined in the data base.
38	MAINREN	OLD NAME NOT IN D.B. Attempt to change name of some object which is not defined in the data base. Probably a spelling error.
39	MAINREN:	NEW NAME ALREADY IN D.B. Attempt to change old name to a name already defined in the data base. User must choose another name.
40	CLREN:	MUST GIVE OLD AND NEW, OR INPUT Parameters given for the command do not supply sufficient information for processing. Reissue command.
41	MAINDEL:	NAME TO BE DELETED NOT IN D.B. Attempt to delete a name which is not defined in the URA data base.
42	MAINDEL:	INVALID MEMBER TYPE URA software error. Please notify persons maintaining URA should this error occur.
43	OTHERS:	CARDINALITY ALREADY GIVEN AS SYSPAR Attempt to assign a numerical value to a CARDINALITY statement when previously assigned a SYSTEM-PARAMETER name. The value is ignored.
44	OTHERS:	CARDINALTIY ALREADY GIVEN AS DIFFERENT VALUE Attempt to assign a second value to a CARDINALITY statement. The new value is ignored.
45	CLCT:	NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER No new name type has been specified. The command must be reissued.
46	CT:	NO NAME GIVEN No name has been specified to have its name type changed. The NAME or FILE parameter must be given.
47	MAINNG:	PD NOT FOUND IN DATA BASE The user specified by the PD parameter is not defined in the data base. No names will be generated.
48	MAINNG:	KEYWORD NOT FOUND IN DATA BASE - The keyword specified by the KEYWORD parameter is not defined in the data base. No names will be denerated.
49	MAINNG:	NO NAMES IN DATA BASE No names have been defined.

AD-A041 825 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370--ETC(U)
APR 75 C R MOORE, H J EIDEN

UNCLASSIFIED ESD-TR-75-88-VOL-3 NL

3 OF 5
AD
A 041825

Number	Subroutine	Error Message
50	PLIST:	TOO MANY NAMES - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
51	RWLIST:	MUST BE SUBSETTING CRITERION NAME Attempt to define a name which is not a GROUP or ELEMENT to be SUBSETTING CRITERION.
52	IDENTC:	NAME NOT IN D.B. - Attempt to retrieve information about a name which is not defined in the data base.
53	OPTRW:	NAME LIST TOO LONG - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
54	OPTRW:	NAME LIST TOO LONG - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
55	OPTRW:	NAME LIST TOO LONG - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
56	USEDTO:	TOO MANY NAMES - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
57	USEDTO:	TOO MANY NAMES - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
58	OPTRW:	NAME LIST TOO LONG - REST IGNORED Exceeded 50 name limit. Remaining names should be given in another statement.
59	MAINNG:	SUBPARTS-OF NAME NOT IN DATA BASE - Attempt to retrieve names that are a part of a name not defined in the data base.
60	APPLES:	SECOND MAILBOX FOR PD ILLEGAL Attempt to associate a second MAILBOX to a particular USER.
61	RWLIST:	ALREADY PART OF SOMETHING ELSE Attempt to define a structure where an object is PART OF more than one other object. This is contrary to the rules specified in ESD TR # 75-88, Vol II .
62	RWLIST:	SECOND PD FOR THIS ITEM ILLEGAL Attempt to assign a second RESPONSIBLE-USER to an object. This statement is ignored.

Number	Subroutine	Error Message
63	RWLIST:	ALREADY PART OF SOMETHING ELSE Attempt to define a structure where an object is PART OF more than one other object. This is contrary to rules specified in ESD TR # 75-88, Vol II .
64	MAINDICT:	NAME NOT FOUND IN D.B. Attempt to retrieve information about name that is not defined in the data base.
65	MAINCONT:	NAME NOT FOUND IN D.B. - Attempt to retrieve information about name that is not defined in the data base.
66	MAINPIC:	NAME NOT IN D.B. - Attempt to retrieve information about name that is not defined in the data base.
67	MAINPIC:	PICTURE NOT AVAILABLE FOR - Attempt to generate a report for a name which is not a SET, INPUT, OUTPUT, ENTITY, GROUP, ELEMENT, PROCESS or REAL-WORLD-ENTITY.
68	REPSET:	WARNING - MISSING SEMICOLON. NEW COMMENT ENTRY ADDED Semicolon not given to terminate comment entry. One is assumed and processing continues.
69	REPSET:	NO NEW COMMENT ENTRY - OLD ENTRY HAS BEEN DELETED Since no new comment entry has been given to replace the old, the old comment entry statement is deleted.
70	CLDCOM:	NO NAME OR FILE SPECIFIED Either the NAME or FILE parameter must be given for this command to be implemented.
76	MAINPRIO:	NAME NOT IN DATA BASE - Attempt to retrieve information about a name that is not defined in the data base.
87	CLDEL:	NO NAME OR FILE WAS SPECIFIED. Either the NAME or FILE parameter must be given for this command to be implemented.
88	MAINPRIO:	NAME NOT A PROCESS NAME - Attempt to retrieve information from a name that is not a PROCESS name.
89	MAINNG:	NAME MUST BE INPUT, OUTPUT, PROCESS, OR RWE FOR "SO" PARAMETER Attempt to retrieve SUBPARTS information for a name whic is not an OUTPUT, INPUT, PROCESS or REAL-WORLD-ENTITY.

Number	Subroutine	Error Message
90	RWLIST:	SSCN IS ONLY LEGAL TYPE IN DEFINE SECTION WHICH CAN BE MAINTAINED Attempt to use MAINTAINED statement for some object which is not SUBSETTING-CRITERION.
91	ADDUSE:	TOO MANY USAGES URA software error. Please notify persons maintaining URA should this error occur.
92	MAINPAV:	NAME NOT IN D.B. - Attempt to retrieve information about name which is not defined in the data base.
93	MAINPAV:	NAME HAS NO USAGES AS ATTRIBUTE FOR ANYTHING - Attempt to retrieve ATTRIBUTE information for a name which is not an ATTRIBUTE.
94	INSYNU:	ERROR OPENING DATA BASE Attempt to access a data base file which is inconsistent. If a newly created file be sure to $COPY SEDJ:URAINITDB TO data-base-file. (See Section 10.1 of this paper).
95	CLEI:	MUST GIVE EITHER ENTITY OR IDENTIFIER PARAMETER Either the ENTITY or IDENTIFIER parameter must be used in conjunction with this command for successful implementation.
98	CONCOL:	NAME NOT IN D.B. - Attempt to retrieve information about a name not defined in the data base.
98	IDENTR:	NAME NOT IN D.B. - Attempt to retrieve information about a name not defined in the data base.
100	ATLIST:	TOO MANY ATTRIBUTE VALUE PAIRS IN SINGLE STATEMENT Limit exceeded. Remaining pairs should be given in another statement.
101	ATLIST:	NAME MUST BE ATTRIBUTE NAME Attempt to use a name defined as something else as an ATTRIBUTE name.
102	ATLIST:	NAME MUST BE ATTRIBUTE VALUE NAME Attempt to use a name defined as something else as an ATTRIBUTE-VALUE name.

Number	Subrout ne	Error Message
103	DELSET:	DESCRIPTION COMMENT ENTRY NOT FOUND FOR ; Attempt to delete a nonexistent DESCRIPTION statement.
104	DELSET:	PROCEDURE COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent PROCEDURE statement.
105	DELSET:	VOLATILITY COMMENT NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY statement.
106	DELSET:	VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY-MEMBER statement.
107	DELSET:	VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY-SET statement.
108	DELSET:	DERIVATION COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent DERIVATION statement.
109	DELSET:	TRUE WHILE COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent TRUE WHILE statement.
110	DELSET:	FALSE WHILE COMMENT NOT FOUND FOR : Attempt to delete a nonexistent FALSE WHILE statement.
111	MAINDCOM:	NAME NOT FOUND IN D.B. : Attempt to delete information for a name not defined in the data base.
113	CLCM:	MUST GIVE EITHER CONSISTS OR CONTAINED PARAMETER Either the CONSISTS or CONTAINED parameter must be used in conjunction with this command.
114	VLIST:	ONLY SINGLE VALUE OR RANGE ALLOWED - IGNORED Invalid format for specifying a VALUES statement. See ESD TR # 75-88, Vol II for the correct format.
115	VLIST:	MIN NOT LESS THAN MAX - IGNORED If a number range is specified the first number must be less than the second number.
116	OTHERS	VALUES ONLY LEGAL FOR ELEMENT, SYSPAR, OR ATTRIBUTE-VALUE Attempt to use a VALUES statement for a name which is not an ELEMENT, SYSTEM-PARAMETER or ATTRIBUTE-VALUE.
117	OTHERS:	DIFFERENT VALUES ALREADY GIVEN Attempt to assign a second value for a given object. This statement is ignored.

Number	Subroutine	Error Message
118	SYSNL:	INVALID SYSPAR GIVEN Error encountered in using a SYSTEM-PARAMETER in a given statement. Interpretation of rest of statement becomes confused.
119	SYSNL:	SYSPAR MUST BE GREATER THAN ZERO Attempt to use zero as a SYSTEM-PARAMETER.
120	SNAMET:	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name in a context which conflicts in the way it has previously been used.
121	MAINRCOM:	NAME NOT FOUND IN DATA BASE - Attempt to access information for a name not defined in the data base.
122	MAINRCOM:	INVALID TYPE OF COMMENT ENTRY - Attempt to replace unrecognizable comment entry statement. Probably a spelling error.
123	MAINRCOM:	CANNOT HAVE THIS TYPE OF COMMENT ENTRY - Attempt to assign a comment entry statement which is not legal for the particular name type.
124	REPSET:	...WITH THIS NAME - Used in conjunction withURA 123. Specifies the name for which the comment entry was used.
125	MAINRCOM:	PROBLEMS SCANNING INPUT FILE - MUST ABORT Incorrect format of file used for input. See command description for correct format.
126	REPSET:	WARNING - THERE IS NO COMMENT ENTRY TO DELETE Attempt to delete nonexistent comment entry.
127	PUNSET:	DESCRIPTION COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent DESCRIPTION statement.
128	PUNSET:	PROCEDURE COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent PROCEDURE statement.
129	PUNSET:	VOLATILITY COMMENT NOT FOUND FOR : Attempt to retrieve nonexistent VOLATILITY statement.
130	PUNSET:	VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent VOLATILITY-MEMBER statement.
131	PUNSET:	VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent VOLATILITY-SET statement.

Number	Subroutine	Error Message
132	PUNSET:	DERIVATION COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent DERIVATION statement.
133	PUNSET:	TRUE WHILE COMMENT ENTRY NOT FOUND FOR : Attempt to retrieve nonexistent TRUE WHILE statement.
134	PUNSET:	FALSE WHILE COMMENT NOT FOUND FOR : Attempt to retrieve nonexistent FALSE WHILE statement.
135	MAINPCON:	NAME NOT FOUND IN D.B. : Attempt to retrieve information for a name not defined in the data base.
141	CONROW:	NAME NOT IN D.B. - Attempt to retrieve information for a name not defined in the data base.
142	BETWEN:	THE TWO NAMES ARE NOT CONNECTED IN THAT FASHION Attempt to delete a relationship, between two names, which is not defined in the data base.
143	MAINDP:	NAME NOT IN D.B. - Attempt to retrieve information about a name which is not defined in the data base.
144	DPCOL:	TOO MANY COLUMNS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
145	DPCOL:	TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
146	DPCOL:	SPARSE MATRIX SYSTEM OVERFLOW Exceed limits of the software that produces the matrix.
147	SPROW:	TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
148	DPROW:	TOO MANY COLUMNS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.
149	DPROW:	SPARSE MATRIX SYSTEM OVERFLOW Exceed limits of the software that produces the matrix.

Number	Subroutine	Error Message
150	DPSUM:	NO ROWS No relationships can be specified about the names used as input, so no matrix will be generated.
151	DPSUM:	NO COLUMNS No relationships can be specified about the names used as input, so no matrix will be generated.
152	DPSUM:	SPARSE MATRIX SYSTEM OVERFLOW Exceeded limits of the software that produces the matrix.
153	MAINDP:	INVALID INPUT NAME TYPE - Attempt to use a name which is not a PROCESS name as input to the command.
154	MAINDP:	INVALID INPUT NAME TYPE - Attempt to use a name which is not a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT name as input to the command.
155	DPSUM:	INVALID ROW TYPE - SYSTEM ERROR URA software error. Please notify persons maintaining URA should this error occur.
156	CNTAND:	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
157	BETWEN:	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
158	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
159	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
160	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which has not been defined in the data base.
161	UDDERS:	NO CONNECTIVITY EXISTS Attempt to delete a relationship which does not exist for this name.
162	UDDERS:	DIFFERENT CONNECTIVITY IN DATA BASE - NOT DELETED Attempt to delete a relationship which is not stated exactly as is in the data base.

Number	Subroutine	Error Message
163	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
164	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
165	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
166	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
167	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
168	UDDERS:	DIFFERENT VALUES IN DATA BASE - NOT DELETED Attempt to delete a number or range of numbers that was not defined for the statement.
169	UDDERS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
170	DELSYN:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
171	DELSYN:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
172	DELSYN:	NAME IS NOT A SYNONYM FOR THIS NAME Attempt to delete a SYNONYM relationship which is not defined in the data base.
173	DRIVES:	USES INFORMATION NOT IN DATA BASE Attempt to delete a relationship which is not specified exactly as in the data base. Relationship is not deleted.

Number	Subroutine	Error Message
174	DRIVES:	WARNING - "USING" INFO IN DATA BASE Statement deleted although the relationship has not been specified exactly as in the data base (the "USING" clause has been omitted).
175	DRIVES:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
176	DRIVES:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
177	DRIVES:	THESE TWO NAMES NOT CONNECTED IN THAT WAY Attempt to delete a relationship which is not defined in the data base.
178	HAPPNS:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
179	DRIVES:	"USES" NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
180	HAPPNS:	NAMES NOT CONNECTED Attempt to delete a relationship which is not defined in the data base.
181	DISCON:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
182	DISCON:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
183	DISCON:	NAMES NOT CONNECTED Attempt to delete a relationship which is not defined in the data base.
184	ATVLST:	NAME DOESN'T HAVE THIS ATTRIBUTE Attempt to delete a relationship which is not defined in the data base.
185	ATVLST:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.

Number	Subroutine	Error Message
186	ATVLST:	NAME HAS NO ATTRIBUTES Attempt to delete ATTRIBUTE relationship for a name with no ATTRIBUTES.
187	NOCNST:	SYSPAR VALUE IN DATA BASE IS DIFFERENT - IGNORED Attempt to delete a statement using a different SYSTEM-PARAMETER. Statement not deleted.
188	NOCNST:	WARNING - SYSPAR IN DATA BASE Statement deleted though user did not include SYSTEM-PARAMETER with statement.
189	NOCNST:	NAME NOT IN DATA BASE Attempt to delete a CONSISTS relationship using a name not defined in the data base.
190	NOCNST:	CONSISTS/CONTAINED INFORMATION NOT IN DATA BASE Attempt to delete a CONSISTS or CONTAINED relationship not defined in the data base.
191	NDCNST:	NO SYSPAR IN DATA BASE - IGNORED Attempt to delete a relationship which is not defined exactly in the same way as defined in the data base. Statement not deleted.
192	CONN:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
193	CONN:	RELATION HAS NO CONNECTIVITY Attempt to delete a CONNECTIVITY relationship for a name with no CONNECTIVITY statements associated with it.
194	CONN:	DIFFERENT CONNECTIVITY IN DATA BASE Attempt to delete a CONNECTIVITY relationship not defined exactly as is in the data base. Statement not deleted.
195	SYSVAL:	NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.
196	PLONG:	COMMENT NOT FOUND *** SYSTEM ERROR *** URA software error. Please notify persons maintaining URA should this error occur.
197	COMNT:	COMMENT-ENTRIES NOT ALLOWED IN DURL Attempt to delete comment-entry statements. This can only be done using the DCOM and RCOM commands. Statement not deleted.

Number	Subroutine	Error Message
198	COMNT:	EOF WHILE LOOKING FOR SEMICOLON Improper statement syntax has been encountered. Semicolons are needed to end all URL statements.
199	HAPPNS:	DIFFERENT SYSPAR - NOT DELETED Attempt to delete a HAPPENS relationship using a different SYSTEM-PARAMETER than defined in the data base. Statement not deleted.
200	HAPPNS:	INTERVAL NOT IN DATA BASE Attempt to delete a HAPPENS relationship using an INTERVAL not defined in the data base.
201	PLIST:	NAME NOT PART OF HEADER An illegal statement header has been given. Probably a spelling error. The statement is ignored.
202	NLIST:	NAME PREVIOUSLY USED DIFFERENTLY - IGNORED Attempt to use a name in a context different than the way it is defined.
203	DRIVES:	"USING" NAME NOT IN DATA BASE Attempt to delete a relationship not defined in the data base.
204	DEFN:	INVALID NAME TYPE Attempt to assign a name type to a name which is used in a different context.
205	SETSYN:	ALREADY SYNONYM FOR SOMETHING ELSE Attempt to assign a name to be a SYNONYM for more than object.
206	SETSYN:	UNABLE TO MAKE SYNONYM - TOO COMPLICATED See Section 10.1 for explanation and solution to this error.
207	SETSYN:	CANNOT BE MADE SYNONYM - DIFFERENT TYPES Attempt to assign a name as a SYNONYM to a different name, both with different name types.
209	SYSNL:	NAME MUST BE INTERVAL Attempt to use a name which is not an INTERVAL in a CONSISTS statement for an INTERVAL section.
210	SYSNL:	INVALID NAME TYPE Attempt to use a name in a context different than way name is defined.

Number	Subroutine	Error Message
211	OTHERS:	NAME MUST BE ENTITY NAME Attempt to use a name in a context where only an ENTITY name is acceptable.
212	OTHERS:	RELATION ALREADY EXISTS BETWEEN TWO OTHER ENTITIES Attempt to specify the same RELATION for a different pair of ENTITIES. Different ENTITY pairs imply different RELATIONS.
213	OTHERS:	CAN ONLY HAVE ON CARDINALITY Attempt to specify a second CARDINALITY statement for a name.
214	OTHERS:	CONNECTIVITY ALREADY GIVEN FOR THIS RELATION Attempt to specify a second CONNECTIVITY statement for a name.
215	OTHERS:	ALREADY CONTAINS WITH DIFFERENT SYSTEM PARAMETER Attempt to specify the same CONSISTS statement, but with two different SYSTEM-PARAMETERS.
216	OTHERS:	NAME MUST BE ENTITY NAME BEFORE VIA Attempt to use a name in a statement where only an ENTITY name is allowed.
217	OTHERS:	NAME MUST BE RELATION AFTER VIA Attempt to use a name in a statement where only a RELATION name is allowed.
218	OTHERS:	RELATION ALREADY EXISTS BETWEEN DIFFERENT ENTITY PAIR Attempt to specify the same RELATION for a different pair ENTITIES. Different ENTITY pairs imply different RELATIONS.
219	OTHERS:	NAME MUST BE CONDITION Attempt to use a name in a statement where only a CONDITION name is allowed.
221	NLIST:	TOO MANY NAMES - REST IGNORED Attempt to specify a list of names for a statement where only a single name is acceptable.
222	RWLIST:	NAME MUST BE GROUP OR ELEMENT Attempt to use a name in a statement where only a GROUP or ELEMENT name is acceptable.
223	RWLIST:	NAME MUST BE SET, ENTITY, GROUP, ELEMENT, OR INPUT Attempt to use a name in a statement where only a SET, ENTITY, GROUP, ELEMENT, or INPUT is acceptable.

Number	Subroutine	Error Message
224	RWLIST:	IDENTIFIER MUST BE GROUP OR ELEMENT Attempt to use a name in a statement where only a GROUP or ELEMENT name is acceptable.
225	RWLIST:	CANNOT HAVE KEYWORD FOR KEYWORD Attempt to assign a KEYWORD to a KEYWORD name.
226	RWLIST:	ONLY RELATIONS AND SEC'S CAN BE MAINTAINED Attempt to use a URL statement in the wrong context.
227	RWLIST:	PD CANNOT BE RESPONSIBLE FOR PD Attempt to assign a RESPONSIBLE-USER to a USER . name.
228	RWLIST:	CANNOT HAVE SECURITY FOR SECURITY Attempt to assign a SECURITY statement to a SECURITY name.
229	RWLIST:	CANNOT HAVE SOURCE FOR SOURCE Attempt to assign a SOURCE to a SOURCE name.
230	RWLIST:	SSC MUST BE SSC, GROUP, OR ELEMENT Attempt to define a name, which is not a GROUP or ELEMENT, as SUBSETTING-CRITERIA for a SET name.
231	RWLIST:	SYNONYMS ONLY APPLIED TO FIRST NAME Attempt to assign a SYNONYM to more than one name. The SYNONYM is given only to the first name.
232	APPLES:	APPLIES STATEMENT ILLEGAL WITH THIS NAME TYPE Attempt to use APPLIES statement for a name which is not a KEYWORD, MAILBOX, SECURITY or SOURCE.
233	DEFN:	TOO MANY NAMES IN DEFINE HEADER - REST IGNORED Exceeded 50 name limit, remaining names should be given in another statement.
234	OPTRW:	NAME MUST BE PROCESS Attempt to use a name in a wrong context. Only a PROCESS name can be used in this context.
235	OPTRW:	NAME MUST BE ELEMENT, GROUP, ENTITY, OR SET Attempt to use a name in wrong context for an UPDATES relationship.
236	OPTRW:	MUST BE ELEMENT, GROUP, INPUT, ENTITY, OR SET Attempt to use a name in wrong context for a USES or USING relationship.

Number	Subroutine	Error Message
237	USEDTO:	MUST BE PROCESS NAME Attempt to use a name in a wrong context. Only a PROCESS name can be used in this context.
238	USEDTO:	MUST BE ELEMENT, GROUP, ENTITY, OUTPUT, OR SET Attempt to use a name in the wrong context for a DERIVES relationship.
239	USEDTO:	MUST BE ELEMENT, GROUP, ENTITY, OR SET Attempt to use a name in the wrong context for an UPDATE relationship.
240	APPLES:	KEYWORD CANNOT APPLY TO KEYWORD Attempt to use the APPLIES statement in the wrong context.
241	APPLES:	MAILBOX CAN ONLY APPLY TO PD Attempt to use the APPLIES statement in the wrong context.
246	APPLES:	SECURITY CANNOT APPLY TO SECURITY Attempt to use the APPLIES statement in the wrong context.
247	APPLES:	SOURCE CANNOT APPLY TO SOURCE Attempt to use the APPLIES statement in the wrong context.
248	APPLES:	MEMO CANNOT APPLY TO MEMO Attempt to use the APPLIES statement in the wrong context.
249	APPLES:	INVALID SECTION - WOOPS URA software error. Please notify persons maintaining URA should this error occur.
251	SNAMET:	ATTEMPT TO CHANGE TYPE WHEN ALREADY TYPED URA software error. Please notify persons maintaining URA should this error occur.
252	SETSYN:	SYNONYM TABLE OVERFLOW Exceeded URA limits. The user should reissue INPUT-URL command at point in the input file where this error occurred.
253	RWLIST:	INVALID STATEMENT NUMBER URA software error. Please notify persons maintaining URA should this error occur.
254	NAMDBK:	CANNOT CREATE SYNONYM URA software error. Please notify persons maintaining URA should this error occur.
256	CONTND:	INVALID SECTION URA software error. Please notify persons maintaining URA should this error occur.

Number	Subroutine	Error Message
257	CONTND:	MUST BE GROUP, INPUT, OUTPUT, OR ENTITY Attempt to use the CONTAINED relationship in a wrong context.
258	CONTND:	MUST BE SET Attempt to use the CONTAINED relationship in a wrong context. INPUTS, OUTPUTS and ENTITIES can only be CONTAINED in a SET.
259	CONTND:	MUST BE GROUP, INPUT, OUTPUT, OR ENTITY Attempt to use a name in the wrong context for a CONTAINED relationship.
263	OPTRW:	MUST BE ELEMENT, GROUP, OUTPUT, ENTITY, OR SET Attempt to use a name, defined to be something else, as a SYSTEM-PARAMETER.
264	SYNTH:	NAME MUST BE SYSPAR Attempt to use a name, defined to be something else, as a SYSTEM-PARAMETER.
265	HAPENS:	SAME THING, SAME INTERVAL, DIFFERENT SYSPAR Attempt to specify same relationship though different SYSTEM-PARAMETER. Not allowed.
266	ILLST:	ILLEGAL STATEMENT IN THIS SECTION Attempt to use a URL statement in a wrong context. See ESD TR # 75-88, Vol II for where this statement may be used.
267	ILLST:	NO CURRENT SECTION Attempt to use an illegal section header statement. See ESD TR # 75-88, Vol II for list of legal sections.
268	USEDTO:	NAME MUST BE SET, ENTITY, GROUP, ELEMENT, OR INPUT Attempt to use a name in the wrong context for a USES relationship.
269	NLIST:	NAME LIST TOO LONG, REST IGNORED Limit of 50 names has been exceeded. Remaining names should be given in another statement.
270	INPAR:	ERROR OPENING DATA BASE - MUST ABORT Attempt to use an inconsistent data base. No processing can be done on it.
271	MAINCNC:	NAME NOT IN D.B. - Attempt to retrieve information for a name not defined in the data base.

Number	Subroutine	Error Message
272	CNCBLD:	TOO MANY ROWS - STOPPING HERE Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CNC command.
273	CNCBLD:	NAME DOESNT CONSIST OF ANYTHING - No information can be presented for this name in the matrix.
274	CNC LD:	TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED Too many levels of CONSISTS information to be presented.
275	CNCBLD:	***THE FOLLOWING NAMES ARE INVOLVED IN A LOOP: This problem should be corrected by modifying the CONSISTS statements for these names.
276	CNCBLD:	TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED Too many levels of CONSISTS information to be presented.
277	CNCBLD:	TOO MANY COLUMNS - STOPPING HERE Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CNC command.
278	CNCBLD:	SPARSE MATRIX OVERFLOW - STOPPING HERE Exceeded limits of software that produces the matrix.
279	CNCSUM:	***NO COLUMNS, OR NO ROWS - STOPPING No relationships can be specified about the names used as input, so no matrix will be generated.
280	CNCSUM:	LESS THAN 2 ROWS, NO SIMILARITY MATRIX Not enough information is available to generate a matrix.
281	CNCSUM:	SPARSE MATRIX OVERFLOW - STOPPING Exceeded limits of software that produces the matrix.
301	IDENTR:	***TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another EI command.
302	IDENTR:	***TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another EI command.
303	IDENTR:	***MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces the matrix.

Number	Subroutine	Error Message
304	IDENTR:	THE FOLLOWING NAMES DO NOT IDENTIFY ANYTHING: No information can be presented in the matrix for these names.
305	IDENTC:	*** TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another EI command.
306	IDENTC:	*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another EI command.
307	IDENTC:	*** MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces the matrix.
308	IDENTC:	THE FOLLOWING NAMES ARE NOT IDENTIFIED BY ANYTHING: No information can be presented in the matrix for these names.
309	CONROW:	*** TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CM command.
310	CONROW:	*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CM command.
311	CONROW:	THE FOLLOWING ARE NOT CONTAINED IN ANYTHING: No information can be presented in the matrix for these names.
312	CONROW:	*** MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces this matrix.
313	CONCOL:	*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CM command.
314	CONCOL:	*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CM command.
315	CONCOL:	THE FOLLOWING DO NOT CONSIST OF ANYTHING: No CONSISTS statements have been used in conjunction with the names listed.

Number	Subroutine	Error Message
316	CONCOL:	*** MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces this matrix.
317	CHKREL:	NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name in a context different than initially defined and used.

10. How to Correct Errors

Once error diagnostics are generated, there must be some method to deal with them. When the errors are caused by problems in generating a report, no action need be taken as no harm will come to the data base. The Report Command can simply be restated in correct format to solve the problem. If, however, an error is encountered in making modifications to the data base (via Modifier Commands) then some immediate action should be taken if the user desires to maintain a correct and complete user requirement.

The errors discovered in making modifications to the data base can be "Input Errors" which are errors discovered by URA in its attempt to process the information needed to update the data base. All these errors are specified by one or more URA error messages and in ninety percent of the cases these errors occur in the process of using the INPUT-URL command.

The errors discovered in the user requirement by the user are called "Logical Errors." No error diagnostics were generated by URA to denote that an error had occurred. If a name was misspelled in the input information used for INPUT-URL, the name could be legal by URL/URA conventions yet not correct from the user's standpoint. "BATCH" and "BATHC" would be perfectly acceptable to URA but not to the user.

The following two sections deal with aiding the user in correcting both Input Errors and Logical Errors should they occur. Treatment of the methodology is still at a high level and no attempt is made to present procedures of correcting all possible errors.

10.1 Input Errors

As stated before, all input errors cause URA error diagnostics to be printed. They are a few classes of errors which happen again and again and so will be described below.

Inconsistent Data Base

This error is usually identified by getting the URA error: "URA094: INSYNU: ERROR OPENNING DATA BASE." You might get this error after issuing a URA Modifier or Report Command and it specifies that the contents of the file you are using as a URA data base cannot be accessed by the URA software. There are several possible reasons for getting this error:

i) The user has attempted to use a dataset as a database file, i.e., the dataset being used in the SET DB command was not created and initialized as a URA database. See Section 2 of this paper for requirements for a database. If this is the problem and the user is still in URA mode, the following sequence of commands should be given:

```
! (ATTN Interrupt)

DELETE dataset-name.data

EXEC NEWDB 'dataset-name.database'

EXEC CLI

SET DB=dataset-name.database
```

where 'dataset-name' is the name of the database to be accessed.

ii) If the error is not a result of the above, and information has been previously stored in the database, then there is probably no way to recover it unless the user has made a copy of the database by issuing:

```
COPY dbname.database SAVE.DATABASE
```

if an inconsistency error has been encountered, the user may restore the database to the state it was in when the COPY command was issued by:

```
DELETE database-name.database

COPY SAVE.DATABASE database-name.database
```

The "SAVE.database" may then be destroyed if no longer needed. This error can occur if TSO crashes or information from another dataset is copied into the database file.

URL Statement Error

These errors account for ninety percent of the errors encountered when inputting information into the database via INPUT-URL. These errors are caused by improper use of URL statements according to the rules specified in the URL Users Manual (ESD/MCI TR # XXXXXXXXXX). An occurrence of any of these errors result in the statement, where the error occurred to be ignored by the system.

The "$" character printed by URA is usually fairly close in pointing out where the error occurred. Some of the more common errors (and solutions) are presented here in hopes that the users will be able to apply the methods of solving these errors to their own, specific needs.

i) Syntax Errors

These errors are often encountered through misspellings, improper format of the statement or improper usage of PSL reserved words. URA usually generates either of the two error messages:

```
URA010:REDUCE:  NO APPLICABLE PRODUCTION-SYNTAX ERROR-START SKIPPING
```

or,

```
URA011:STACK:  ILLEGAL SYMBOL PAIR-SYNTAX ERROR-START SKIPPING
```

For example, if you misspell the RECEIVES statement:

```
RECIEVES  FOLDER-A,FOLDER-B;
```

URA will react with the URA010 error message and skip that statement to go on to the next. There are some further problems that can occur then. If the error occurred in a header statement, such as PROCESS, etc., then the header statement is skipped and all statements intended to be related to the header statement will be related to the previous header statement. Take the example:

```
GROUP:  G1;
CSTS:  E1, E2, G2;
PROCCESS:  P1;
RCVS:  I1, I2;
SUBPARTS  P2, P3;
ELEMENT  E1, E2;
```

PROCESS has been misspelled which results in having that header statement skipped. All the statements following this header are related to the previous header which leads to more problems since statements which can only be associated to a PROCESS are being given to a GROUP name. More errors will occur from this resulting in the PROCESS, RCVS and SUBPARTS statements not being entered into the data base. To correct this error, the statements that were omitted could be entered by another INPUT-PSL command. A far more serious problem occurs if the "previous" header was also a PROCESS in our particular example:

```
PROCESS PX;
RCVS:  I1,I3;
GENS:  O1,O2;
PROCCESS  P1;
RCVS  I1, I2;
SUBPARTS  P2,P3;
ELEMENT  E1,E2;
```

If this were the case, then only one error would be caught (the misspelled "PROCCESS") and the trailing RCVS and SUBPARTS statements would be given to PROCESS PX. If this mistake was discovered, the user would have to delete the two statemerts from PX and then reinput the information for P1:

```
DELETE-URL
PROCESS PX;
RCVS  I1,I2;
SUBPARTS  P2,P3;
EOF
INPUT-URL  UPDATE
PROCESS  P1;
RCVS  I1,I2;
SUBPARTS  P2,P3;
EOF
```

ii) Illegal Statement

This error is designated by the PSA error:

```
URA266:ILLST:  ILLEGAL STATEMENT IN THIS SECTION
```

This error can be caused by simply using a statement that is not allowed for that particular section. Using a CONSISTS statement in a PROCESS section would obviously generate this error. The other case occurs when an error is made in a header section statement and all the following statements might be incompatible with the previous header section name. Note that whenever this error is encountered, the statement is not put into the data base.

iii) Illegal Header Statement

If an error occurs in a header statement and PSA is able to identify it as a header statement, the following error will be given:

```
URA266:ILLST:  INVALID HEADER STATEMENT-STATEMENTS WILL BE IGNORED
```

This means that all the statements up to the next header statement will be ignored and not put into the data base. All the statements ignored must be reinputted to another command to be put into the data base.

iv) Input Line Too Long

If a number of URL statements are used on one line of the input file or URL statement is very long, it may run over the 72 column restriction. Should this occur, usually URA010 or URA011 will be generated specifying that improper syntax has been countered. Note that no error message is generated for the fact that the statement runs over the 72 column restriction. Errors are encountered because anything over column 72 is ignored no matter what type of character it is and so names may be truncated or a semicolon is missing.

v) Name Too Long

Since there is a thirty character limit in forming names, it is sometimes the case that we think 30 characters to be a lot longer than it really is. Instead our name may be 32 or 33 characters long and this is caught by URA and flagged by the error:

```
URA002:NLEX:  NAME TOO LONG
```

The statement that used the name is still entered into the data base but the name is stored in truncated form in the data base. If the truncated form of the name is not satisfactory, it is a simple matter to change the name via the RENAME command.

vi) Using URA Reserved Words Incorrectly

Most syntax errors are fairly easy to detect; a misspelled word, improper format, etc., but one of the hardest to detect is the improper use of a URL reserved word. For example, the following statement would be flagged as having a syntax error.

```
ATTRIBUTE  TYPE  A;
```

The letter "A" happens to be a URL optional word and cannot be used as a user-defined name. Detecting these reserved words can get trickier than this, however, as the statement:

```
PROCESS D,F,G,K;
```

seems all right, though "F" is the abbreviation to the URL reserved word "FALSE." The key to finding these errors is to watch where the "$" character is printed by URA. It is usually printed directly after the location of the source of the error. The statement is ignored should this type of error occur and the only solution is to reinput the data using a different name.

vi) Synonym Too Complicated

This error is specified by the URA error:

```
URA206:SETSYN:  UNABLE TO MAKE SYNONYM-TOO COMPLICATED
```

This is caused by specifying various relationships about two names and then attempting to make one a SYNONYM of the other. The problem lies in combining these relationships. The statements:

```
GROUP  G1;
USED BY  P2;
PROCESS LONG-PROCESS-NAME;
SUBPARTS  P3,P4;
SYNONYM  P2;
```

will generate the error. P2 is implicitly defined to be a PROCESS just in the context that it is used in the second statement and also have a relationship with G1; now LONG-PROCESS-NAME is defined and has relationships formed with P3 and P4. In the last statement, an attempt was made to make P2 and LONG-PROCESS-NAME the same PROCESS and the error is generated. The whole problem could have been avoided if the user maintained the convention of issuing SYNONYM statements directly after the header statement as shown below:

```
GROUP  G1;
USED BY  P2;
PROCESS  LONG-PROCESS-NAME;
SYNONYM  P2;
SUBPARTS  P3;P4;
```

If the statements had been inputted in this manner, LONG-PROCESS-NAME would not have had any relationships formed with other names (P3 and P4 in the previous example) and the assignement of P2 as a SYNONYM would have been successful.

Since the error does occur, there exists a method of correcting this problem. First, all information about one of the names, P2 or LONG-PROCESS-NAME,must be retrieved and then that name deleted from the data base. The information can then be editted to be acceptable by URA and reinputted via INPUT-URL.

```
      FPS N=P2  P=PUNCH
      DELETE N=P2
!
EDIT PUNCH
CHANGE 1 /P2/LONG-PROCESS-NAME/
1.5 SYNONYM P2;
SE
EXEC CLI
      INPUT-URL I=PUNCH U
```

In this way, all information about P2 is given to LONG-PROCESS-NAME and P2 is assigned as a SYNONYM if future references to the name are necessary. Really, it would have been much easier if you maintain the convention of assigning SYNONYMS directly after the header statement.

vii) Names Used in Wrong Context

This type of error accounts for the majority of different error messages presented in Section 9.

```
URA202:NLIST:  NAME PREVIOUSLY USED DIFFERENTLY-IGNORED
```

and,

```
URA222:RWLIST:  NAME MUST BE GROUP OR ELEMENT
```

are examples of diagnostics presented for this type of error. The statement will be ignored and the only way to resolve the problem is to reinput the information in an acceptable format.

viii) Breaking Section/Statement Rules

Several error messages can be generated by attempting to break the rules set forth in ESD TR #   for statements within a particular section. In using the PART statement, for example, an object may be PART of only one object and failure to comply with this rule will result in:

```
URA061:RWLIST:  ALREADY PART OF SOMETHING ELSE
```

or some analogous URA error message. These error checks are made to enforce the rules set forth in Working Paper No 68 and ensure that the user requirement is still meaningful. Other messages presented for this type of error are:

```
URA214:OTHERS:  CONNECTIVITY ALREADY GIVEN FOR THIS RELATION
```

or,

```
URA060:APPLES:  SECOND MAILBOX FOR PD ILLEGAL
```

If the user wishes to replace the information stated in the data base, say replace the MAILBOX for a user, the relationship should be deleted via DELETE-URL and then input the correct information using the INPUT-URL command.

10.2 Logical Errors

These errors occur when inputting information into the data base (as input errors do), but no diagnostics are given in the AS-IS SOURCE LISTING. These errors might be detected by scanning the complete list of names in the data base (NAME-GEN) and the complete user requirement (FORMATTED-PROBLEM-STATEMENT). These errors can also be detected when reviewing the contents of any of the other reports available on URA.

Misspelled Names

A simple spelling error can result in two names which look very similar, but which are treated as two different objects in the data base.

i) For example, if we used the name, "CALENDAR-DAY" to specify a particular INTERVAL in the data base and then used "CALENDAR-DAYS" in the statements:

```
INTERVAL:  CALENDAR-week;
CONSISTS:  7 CALENDAR  DAYS;
```

the two names are completely different objects (to URA). URA does not know that you have made a spelling error and it is up to you to detect and correct it, which can be done in the following manner:

```
          FPS N=CALENDAR-DAYS P=PUNCH
          DELETE N=CALENDAR-DAYS
     !
     EDIT PUNCH
     CHANGE 1 /CALENDAR-DAYS/CALENDAR-DAY/
     SE
     EXEC CLI
          INPUT-URL I=PUNCH U
```

All information given about CALENDAR-DAYS is transferred to CALENDAR-DAY and then CALENDAR-DAYS is deleted from the data base. If it is desirable to use plural forms of the names in the data base as SYNONYMS, this can be done in a DESIGNATE statement:

```
INPUT-URL
DESG  CALENDAR-DAYS  SYNONYM  CALENDAR-DAY;
EOF
```

ii) Since names can consist of letters or numbers, another common misspelling error is to substitute the letter "O" with the number "0". It is often very difficult to detect this and so there appears to be two names, spelled exactly the same in the data base. This can be corrected in the same way as the previous problem.

iii) When the spelling error only involves one name (if TIME-CARD was spelled TIMECARD in all instances) then this problem could easily be solved by using the RENAME command:

```
RENAME Ø=TIMECARD  N=TIME-CARD
```

If both TIME-CARD and TIMECARD are defined in the data base, then the same procedure used to change CALENDAR-DAYS must be performed.

Redundant Objects

Another error which occurs quite frequently is to define one object by two different names, not realizing that they are truly representing the same thing. EMPLOYEE-RECORD and EMPLOYEE-DATA may be defined separately in the data base, but represent the same thing. To resolve this redundancy, we must combine the information for these two names. If the two names have the same name type associated with them (say ENTITY), then fewer problems are encountered. If they are of different name types, the CHANGE-TYPE command must be used. Now the names can be defined to be used synonymously with each other by the following procedures.

```
     FPS N=EMPLOYEE-DATA P=PUNCH
     DELETE N=EMPLOYEE-DATA
!
EDIT PUNCH
CHANGE 1 /DATA/RECORD/
1.5 SYNONYM:  EMPLOYEE-DATA
SE
EXEC CLI
     INPUT-URL I=PUNCH U
```

Missing Semicolons

Most often, a missing semicolon can be detected as a syntax error (as described in Section 10.1). There is one particular case where a missing semicolon would not generate any error message:

```
    PROCESS P1;
    DESCRIPTION;
        THIS IS A DESCRIPTION COMMENT ENTRY THAT IS MISSING
        THE SEMICOLON.
    RCVS I1,I2;
    GENS O1,O2;
```

What happens here is that the RCVS statement becomes part of the DESCRIPTION comment entry. A semicolon was omitted in terminating the lines we intended as the comment entry, but URA simply searches for the first semicolon to signify the end of the comment entry. To solve this problem the DESCRIPTION statement must be replaced and the RCVS statement must be added to the data base:

```
        PCOM N=P1 DESC P=PUNCH
    !
    EDIT PUNCH
    DELETE 5
    SE
    EXEC CLI
        RCOM F=PUNCH
        INPUT-URL U
        PROCESS P1;
        RCVS I1, I2;
        EOF
```

Correctness and Completeness

For the most part, it is up to the user to maintain correctness of the user requirement and URA maintains correctness of the data base. The user has the ability to do this through usage of the DELETE-URL and INPUT-URL commands. Completeness can also be determined by the user or improved through use of the INPUT-URL command.

APPENDIX A - URA Special Procedures

This appendix consists of sets of procedures to perform specific operations dealing mainly with modification of the URA database. Procedures are presented to aid in using each of the modifier commands:

CHANGE-TYPE

DELETE

DELETE-COMMENT-ENTRY

DELETE-URL

INPUT-URL

RENAME

REPLACE-COMMENT-ENTRY

and to aid in resolving specific problems due to logical errors.

A few of the conventions used for these procedures are listed below:

1. T.DATA is a dataset which is used to contain information specified by the user. If the dataset is to be used more than once in a given terminal session, the old input lines should be deleted using the DELETE command of EDIT before new input lines are specified.

2. PUNCH.DATA is a dataset used to contain information supplied by the PUNCH output of any URA command. If the dataset is to be used more than one in any given terminal session, URA will write over any information left from previous commands.

3. All the words specified in lower case represent names or information which must be replaced by the users for their particular implementation of the procedure.

4. All words specified in uppercase are information which must be stated exactly as shown (except for substitution of legal abbreviations) for the procedure to work correctly.

5. These procedures assume the user is in URA mode when the procedure is implemented.

6. All URA commands are indented from any TSO commands with which they may appear. This indentation is provided solely for clarity and the spaces should not be used at the terminal.

7. Some of the procedures start with an attention interrupt (signified "!") to return to TSO. In all cases, a FREEALL should be issued following the interrupt. The FREEALL commands are not included in the algorithms for clarity.

Preparing a Database File for Access by URA

```
!

EXEC NEWDB 'dbname.DATABASE'

EXEC CLI

    SET DB=dbname.DATABASE
```

CHANGE-TYPE Command Procedures

1. To change the name type for only one name in the database:

```
CT N=name     TYPE=new-name-type
```

2. To change the name types of several names in the database when the name type are not the same:

```
!

EDIT T.DATA NEW

name1    new-name-type 1

name2    new-name-type 2

.   .

.   .

.   .

namen    new-name-typen

(blank line RETURN)

SE

EXEC CLI

    CT FILE=T.DATA
```

3. To change the name types of several names to the same name type:

```
!

EDIT T.DATA NEW

name1
```

```
.

.

.

namen

(blank line RETURN)

SE

EXEC CLI

     CT FILE=T.DATA TYPE=new-name-type
```

DELETE Command Procedures

1. To delete one name from the database:

```
DELETE N=name
```

2. To delete several names from the database:

```
!
EDIT T.DATA NEW
name1
name2
.
.
.
namen
(blank line RETURN)
SE
EXEC CLI
    DEL F=T.DATA
```

DELETE-COMMENT-ENTRY Command Procedures

1. To delete one type of comment entry for one name in the database:

```
DCOM N=name   comment-entry-parameter
```

2. To delete one type of comment entry for several names:

```
!

EDIT T.DATA NEW

name1

.

.

.

namen

(blank line RETURN)

SE

EXEC CLI

    DCOM F=T.DATA  comment-entry-parameter
```

3. To delete two comment entry types for several names:

```
!

EDIT T.DATA NEW

name1

.
```

```
.

.

namen

(blank line RETURN)

SE

EXEC CLI

    DCOM F=T.DATA  comment-entry-param1 comment
entry-param2
```

DELETE-URL Command Procedures

1. To delete several relationships from a URA database using a dataset as input:

```
!

EDIT T.DATA NEW

     legal URL statements

EOF

(blank line RETURN)

SE

EXEC CLI

DURL I=T.DATA
```

2. To delete several relationships from a URA database on an interactive basis (using the terminal):

```
DURL
    )
    } legal URL statements
    )
EOF
```

INPUT-URL Command Procedures

1. To input information into a URA database using the contents of a dataset as input:*

```
!

EDIT T.DATA NEW legal URL statements

EOF

(blank line RETURN)

SE

EXEC CLI

IP I=T.DATA UPDATE
```

2. To interactively input information into a URA database:

```
IP U
   }legal URL statements
EOF
```

*The user may wish to save T.DATA permanently in this case.

RENAME Command Procedures

1. To change the name of one name in the database:

```
REN O=oldname N=newname
```

2. To change the names of several names in the database:

```
!

EDIT T.DATA NEW

old name1     newname1

old name2     oldname2

'        '

'        '

'        '

oldnamen newnamen

(blank line RETURN)

SE

EXEC CLI

REN I=T.DATA
```

REPLACE-COMMENT-ENTRY Command Procedures

1. To completely change comment entry for one name in the database:

```
!

EDIT T.DATA NEW

name

comment-entry-type
    }comment entry text
                         ;
(blank line RETURN)

SE

EXEC CLI

    RCOM I=T.DATA
```

2. To change a comment entry for a name with an edited version of the same comment entry:

```
PCOM N=name comment-entry-parameter P=PUNCH.DATA

!

EDIT PUNCH.DATA

    edit commands to change text

(blank line RETURN)

SE

EXEC CLI

RCOM I=PUNCH.DATA
```

APPENDIX B - URA Command Abbreviations

This appendix presents all commands, and parameters to these commands, for URA. It also presents the acceptable abbreviations for these commands and parameters on the right hand side. There are several conventions in choosing these abbreviations that may aid the user:

1. For command names that consist of only one word, for example, CONTENTS or PICTURE, the first three or four letters of the name are used as they abbreviation. CONT is the abbreviation for CONTENTS and PIC is the abbreviation for PICTURE.

2. For most command names that consist of more than one word, e.g., CONSISTS-MATRIX or FORMATTED-PROBLEM-STATEMENT, the abbreviation is derived by using the first letter from each word in the command name. This gives us CM for CONSISTS-MATRIX and FPS for FORMATTED-PROBLEM-STATEMENT. This convention is not strictly enforced, however, so that the abbreviations may be more meaningful. For example, DCOM is the abbreviation for DELETE-COMMENT-ENTRY which means more than DCE.

3. Abbreviations are the same for those parameters which are used, in the same way, by several commands. The FILE parameter then, always has an abbreviation, F, allowed no matter which command is using it. Some of the more common parameters are listed below:

Parameter	Abbreviation
FILE	F
NAME	N
INPUT	I
NOPRINT	NP
PUNCH	P

4. Whenever an abbreviation exists for a parameter that has a "NO" prefix, such as NOSYNONYMS or NOPRINT, the abbreviation is always prefixed with "N". For example, NSYN is the abbreviation for NOSYNONYMS and NP is the abbreviation for NOPRINT.

Command Name	Parameters	Abbreviations
CHANGE-TYPE		CT
	FILE	F
	NAME	N
	TYPE	T
CONSISTS-COMPARISON		CNC
	FILE	F
CONSISTS-MATRIX		CM
	FILE	F
	NAME	N
	CONTAINED	CNTD
	CONSISTS	CSTS
CONTENTS		CONT
	FILE	F
	NAME	N
	INDEX	
	NOINDEX	
	LEVELS	
	NCFLAG	
	NONCFLAG	
DATA-BASE-STATISTICS		DBS
	NAMES	
	NONAMES	
	NUBS	
	NONUBS	
	NAMNUBS	
	NONAMNUBS	
	SYNONYMS	SYN
	NOSYNONYMS	NSYN
DATA-PROCESS		DP
	FILE	F
	NAME	N
	DATA	D
	PROCESS	P

Command Name	Parameters	Abbreviations
DELETE		DEL
	FILE	F
	NAME	N
DELETE-COMMENT-ENTRY		DCOM
	DERIVATION	DER
	DESCRIPTION	DESC
	FALSE-WHILE	FW
	PROCEDURE	PRCD
	TRUE-WHILE	TW
	VOLATILITY	VOL
	VOLATILITY-MEMBER	VOLM
	VOLATILITY-SET	VOLS
	FILE	F
	NAME	N
	PRINT	
	NOPRINT	NP
DELETE-PSL		DPSL
	INPUT	I
	SOURCE	S
	NOSOURCE	NS
	XREF	X
	NOXREF	NX
DICTIONARY		DICT
	FILE	F
	NAME	N
	INDEX	
	NOINDEX	
	NUM-SPACE	NS
	DESCRIPTION	DESC
	NODESCRIPTION	NDESC
	KEYWORDS	KEY
	NOKEYWORDS	NKEY
	RESPONSIBLE-PD	RPD
	NORESPONSIBLE-PD	NRPD
	SYNONYMS	SYN
	NOSYNONYMS	NSYN

Command Name	Parameters	Abbreviations
ENTITY-IDENTIFIER		EI
	FILE	F
	NAME	N
	IDENTIFIER	I
	ENTITY	E
FORMATTED-PROBLEM-STATEMENT		FPS
	AMARG	AM
	BMARG	BM
	COMMENT	COM
	NOCOMMENT	NCOM
	CMARG	CM
	DEFINE	DEF
	NODEFINE	NDEF
	DESG	DG
	NODESG	NDG
	EMPTY	
	NOEMPTY	
	FILE	F
	NAME	N
	HMARG	HM
	INDEX	
	NOINDEX	
	NEW-LINES	NL
	NONEW-LINES	NNL
	NEW-PAGE	NPG
	NONEW-PAGE	NNPG
	NMARG	NM
	ONE-PER-LINE	OPL
	SEVERAL-PER-LINE	SPL
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	RNMARG	RM
	SMARG	SM

Command Name	Parameters	Abbreviations
FREQUENCY		FREQ
HELP		
	command-name	
	SHORT	
	LONG	
INPUT-PSL		IP
	DBREF	D
	NODBREF	ND
	INPUT	I
	SOURCE	S
	NOSOURCE	NS
	UPDATE	U
	NOUPDATE	NU
	XREF	X
	NOXREF	NX
KWIC		
	FILE	F
	DIF	
NAME-GEN		NG
	ATTRIBUTE	ATTR
	NOATTRIBUTE	NATTR
	ATTRIBUTE-VALUE	ATTV
	NOATTRIBUTE-VALUE	NATTV
	CONDITION	COND
	NOCONDITION	NCOND
	ELEMENT	ELE
	NOELEMENT	NELE
	ENTITY	ENT
	NOENTITY	NENT
	EVENT	EV
	NOEVENT	NEV
	GROUP	GR
	NOGROUP	NGR

Command Name	Parameters	Abbreviations
NAME-GEN (cont'd)		
	INPUT	INP
	NOINPUT	NINP
	INTERVAL	INT
	NOINTERVAL	NINT
	KEYWORD	KEY
	NOKEYWORD	NKEY
	MAILBOX	BOX
	NOMAILBOX	NBOX
	MEMO	
	NOMEMO	NMEMO
	OUTPUT	OUT
	NOOUTPUT	NOUT
	USER	
	NOUSER	
	PROCESS	PROC
	NOPROCESS	NPROC
	REAL-WORLD-ENTITY	RWE
	NOREAL-WORLD-ENTITY	NRWE
	RELATION	RLN
	NORELATION	NRLN
	SECURITY	SEC
	NOSECURITY	NSEC
	SET	
	NOSET	
	SOURCE	SRC
	NOSOURCE	NSRC
	SUBSETTING-CRITERION	SSCN
	NOSUBSETTING-CRITERION	NSSCN
	SYNONYMS	SYN
	NOSYNONYMS	NSYN
	SYSTEM-PARAMETER	SYSP
	NOSYSTEM-PARAMETER	NSYSP
	UNDEFINED	UNDF
	NOUNDEFINED	NUNDF

Command Name	Parameters	Abbreviations
NAME-GEN (cont'd)		
	BASIC	
	NOBASIC	
	EMPTY	
	NOEMPTY	
	KEY	
	IDENTIFIER	ID
	IDENTIFIER-GROUP	IDG
	IDENTIFIER-ELEMENT	IDE
	NONE	
	ALL	
	ORDER	
	PD	
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	TOTAL	
	SUBLEVEL	SL
	SUBPARTS-OF	SO
NAME-LIST		NL
	SYNONYMS	SYN
	NOSYNONYMS	NSYN
PICTURE		PIC
	DATA	D
	NODATA	ND
	FILE	F
	NAME	N
	FLOW	
	NOFLOW	
	INDEX	
	NOINDEX	
	STRUCTURE	STR
	NOSTRUCTURE	NSTR

Command Name	Parameters	Abbreviations
PRINT-ATTRIBUTE-VALUES		PAV
	FILE	F
	NAME	N
PROCESS-INPUT-OUTPUT		PRIO
	FILE	F
	NAME	N
	DESCRIPTION	DESC
	NODESCRIPTION	NDESC
	PROCEDURE	PRCD
	NOPROCEDURE	NPRCD
	INPUT	INP
	NOINPUT	NINP
	OUTPUT	OUT
	NOOUTPUT	NOUT
	NEW-PAGE	NPG
	NONEW-PAGE	NNPG
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	EMPTY	
	INDEX	
	NOINDEX	
PUNCH-COMMENT-ENTRY		PCOM
	DERIVATION	DER
	DESCRIPTION	DESC
	FALSE-WHILE	FW
	PROCEDURE	PRCD
	TRUE-WHILE	TW
	VOLATILITY	VOL
	VOLATILITY-MEMBER	VOLM
	VOLATILITY-SET	VOLS
	EMPTY	
	NOEMPTY	
	FILE	F
	NAME	N

Command Name	Parameters	Abbreviations
PUNCH-COMMENT-ENTRY (cont'd)		
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
RENAME		REN
	INPUT	I
	OLD	O
	NEW	N
REPLACE-COMMENT-ENTRY		RCOM
	INPUT	I
	PRINT	
	NOPRINT	NP
SET		
	ECHO	E
	INPUT	I
	LINES	L
	OUTPUT	O
	PROMPT	P
STOP		
STRUCTURE		STR
	INDENT	IND
	INDEX	IND
	NOINDEX	
	INPUT	INP
	OUTPUT	OUT
	PROCESS	PROC
	REAL-WORLD-ENTITY	RWE
SUMMARY		SUM

Glossary

comment entry
The text associated with a comment entry statement. DESCRIPTION, PROCEDURE and VOLATILITY are examples of comment entry statements.

comment entry statement
Any URL statement which allows the contents of the statement to be defined by the user (as is narrative description). The DESCRIPTION and PROCEDURE statements are examples of this.

Control Command
A URA command which passes control information to URA. STOP and SET are Control Commands.

conversational mode
Used synonymously with on-line mode. Refers to interactive use of the computer system through a terminal device.

database
Synonymous with "URA database". This is the information stored and retrieved by URA by means of the Modifier and Report Commands.

data object
Any URA name type that represents some form of data. SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS are all data objects described by URL.

device
Input-output peripheral equipment; for example, a card reader, magnetic tape drive, terminal, etc.

dsname
Any legal dataset name that is to be specified by the user.

ESD TR#
"URL User Manual", Version 3.0. This manual specifies and defines all the URL statements processable by URA, their purpose, syntax, etc.

ESD TR#
"An Introduction to Computer Aided Requirements Analysis".

header section
Any of the statements allowed in URL that specify the beginning of a set of statements describing the user defined names specified in this statement. See Appendix F of ESD TR# (   ) for a complete list of all section types.

input file
Any dataset which contains data to be used by URA commands via INPUT or FILE parameters.

mode
Term used synonymously with "processing mode" which refers to a state where the user has a particular set of operations available to perform required tasks.

Modifier Command

Any URA command which alters the contents of the user's URA database.

name type

Any of the many types of names allowed in URL (i.e., PROCESS, SET, GROUP, etc.). See ESD TR# (   ), "URL Users Manual", Appendix E for a list of all possible name types.

Report Command

Any command available on the URA system whose sole purpose is to retrieve information from the user's URA database.

undefined name

A name that has been entered into the URA database, but has no name type asspciated with it.

URA

The User Requirement Analyzer. Synonym is "Analyzer". Software package which retrieves and inputs information to the URA database.

URA command

Any of the commands that can be used to operate URA. See Appendix I for complete descriptions about each command available on URA.

URA database

Area where URL information is stored (in a coded format) which can then be accessed by the commands allowed by URA.

URA "object"

Synonymous with "name types". Any of the objects that can be defined by URL (a SET, a GROUP, a PROCESS are all objects in URA). Usually referred to as just "object" in this paper.

URL

Acronym for "User Requirement Language" which is the collection of all URL statements allowed for use by URA. See ESD TR# (   ) for complete descriptions of all statements available.

URL statements

Those statements specified by ESD TR# (   ). The statement presents one aspect of description for a particular URA "object".

user requirement

A set of requirements specified by users of a proposed system and interpreted by the user into a format acceptable by the organization.

PART II

URA COMMAND

DESCRIPTIONS

Introduction

The objective of this part is to give the user of the User Requirements Analyzer (URA) a list of the commands available to him, the correct syntax of these commands, and the parameters allowable for each command when using URA in conjunction with the TSO. The facilities of URA Version A2.0 are addressed. This part assumes familiarity with the concepts concerning URL/URA as presented in ESD TR 75-88, Vol I, "Introduction to Computer Aided Requirements Analysis".

Format of Command Descriptions

All the URA commands in this paper are described in the following format:

COMMAND NAME

Purpose: This presents the function of the command in the URA system whether it generates a report, modifies the data base or gives control information to URA. (Appendix II, "URA OUTPUTS" presents a detailed description of the reports generated by each command.)

Prototype: This presents the legal syntax for the command. The Usage Rules specify what the special symbols (such as braces and brackets) represent in interpreting the syntax.

Parameters: For each parameter available for the command, this section provides a brief description explaining how the parameter chanoes the action of the command.

Defaults: These present which parameters will be used for the command, or what value a parameter will have, if the parameter, or value, is not explicitly defined. *For example, by specifying:*

```
CONTENTS
```

The Defaults for this command show that this has the same effect as specifying:

```
CONTENTS FILE=URANAMES NOINDEX LEVELS=ALL  NONCFLAG
```

If a "no default" is given, this means that if not explicitly defined, the corresponding parameter will not be used for the command.

Examples: Actual example of the command syntax are presented.

Usage Rules

ABBREVIATIONS

To enable the user to fit a lengthy command on the allotted line and eliminate some of the tedium of command specification, abbreviated forms for both commands and parameters may be used. Each abbreviation can be found in parentheses immediately following the word it represents. For example, the command:

```
CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT     can be written as

CT   N=GROSS-PAY   T=ELEMENT
```

BLANKS

A blank must appear between the command and any accompanying parameter, and between successive parameters. Several blanks are treated as a single blank and may be inserted whenever a single blank is necessary. For example,

```
CT N=GROSS-PAY T=ELEMENT     can be written as

CT   N=GROSS-PAY   T=ELEMENT
```

BRACES

In the following examples, when parameters or parameter values are enclosed in braces ({ }), a choice among the two or more entries must be made. It is important to note that one and only one of the options must be chosen. For example, the braces used in describing the syntax of the CHANGE-TYPE command specifies that the command must either be of the form:

```
    CT   N=user-name   T=name-type
or
    CT   F=dsname      [T=name-type]
```

BRACKETS

Whenever parameter notation in an example appears within brackets ([ ]), it indicates a feature the user may optionally use. For example, the brackets around the TYPE parameter in the CHANGE-TYPE command is optional when the FILE parameter is also used. Therefore, the command may be of the form:

```
CT   FILE=dsname   TYPE=name-type

CT   FILE=dsname
```

The syntax of the FILE parameter shows that the parameter may be given either as:

FILE=dsname   or just

FILE

No other variations are acceptable (except those already specified, i.e., abbreviations, etc.).

COMMAND LINE

Each command must appear on a separate line and totally on that line. A command cannot be split on succeeding lines. Only columns 1 through 80 of each line can be used.

COMMAND PARAMETERS

Parameters for a command are separated by one or more blanks. Parameters may be given in any order, but are processed from left to right. If conflicting parameters are used, the right-most parameter is considered to be correct and is the one used in the processing of the command.

ELLIPSIS

The ellipsis (...) signifies that the command construct immediately preceding the ellipsis can be repeated as many times as desired by the user.

GENERAL STATEMENT SYNTAX

The command identifiers (name of the command) must precede any accompanying parameter or list of parameters.

INTEGERS

The integers required for parameters must be positive integers. If a value range is given for a particular parameter description, that restriction must also be met.

NAMES

All used defined names (user-name) must meet the following restrictions to be a legal URL name.

A name can be formed only from the following characters:

A, B, C, ..., Z (letters)
0, 1, 2, ..., 9 (integers)
              - (dash)

- A name can be any combination of thirty characters or less where the first character is a letter.

- Blanks cannot be used in the name.

- A user defined name cannot be a URL RESERVED WORD. For a list of Reserved words see APPENDIX B of ESD TR# 75-88, Vol II .

For example,

GROSS-PAY, EMPLOYEE-NUMBER and PAYROLL-PROCESSING

are all legal names.

PROCESS, EMPLOYEE-# and 123-HILL-STREET

are illegal names. "PROCESS" cannot be used as a user-name because it is a URL Reserved Word. "EMPLOYEE-#" uses a character (the "#" sign) which is not allowed and finally, "123-HILL-STREET" is illegal because it starts with an integer rather than a letter.

CHANGE-TYPE

Purpose: To change the name type of a user name defined ir the user's data base. A record of this change is generated in the form of the CHANGE-TYPE REPORT.

Prototype:

```
CHANGE-TYPE (CT)  {NAME(N)=user-name  TYPE(T)=name-type  }
                  {FILE(F)[=dsname]  [TYPE(T)=name-type] }
```

Parameters: FILE(F)[=dsname] Default: no default

When the FILE parameter is used and no dsname is designated, the contents of the file, -URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. The file format for each line of the input file must be of the form:

user-name [name-type]

Free format is allowed so the user-name does not have to start in the first position in the line. The two names must be separated by one or more blanks. The name-type is optional. If a name-type is not specified for each user-name, the name type for each of these names will be changed to the type specified in the TYPE parameter. One of these alternative methods of assigning a type must be used, but not both. If both are used, all the names in the file will be assigned the name type specified by the TYPE parameter.

NAME(N)=user-name Default: no default

The given user-name is the name for which the change is to be made. When the NAME parameter is used, the TYPE parameter must be used in conjunction with it.

TYPE(T)=name-type Default: no default

This parameter specifies the new name type to be used in the change. See Appendix E of ESD TR# 75-88, Vol II for a list of all possible name types.

Examples:

```
CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT

CT   F=T.DATA   T=ELEMENT

CT   FILE   T=GROUP
```

CONSISTS-COMPARISON

Purpose: To produce the CONSISTS COMPARISON REPORT.

Prototype: CONSISTS-COMPARISON(CNC) [parameter]...

Parameters: FILE(F)[=dsname] Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. In any case, the names in the input file must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

Examples:

```
CNC

CNC   F=T.DATA
```

CONSISTS-MATRIX

Purpose: To produce the CONSISTS MATRIX REPORT.

Prototype: CONSISTS-MATRIX(CM) $\left\{\begin{array}{l}\text{CONTAINED(CNTD)}\\ \text{CONSISTS(CSTS)}\end{array}\right\}$ [parameters]...

Parameter: FILE(F)[=dsname], NAME(N)=user-name **Default:** FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated that file is used as the input file for the command. When a name is specified via the NAME parameter, the report is produced only for that name. The format of the input file must be one name per line.

CONTAINED(CNTD), CONSISTS(CSTS) **Default:** no default

Since no default exists, one of the above must be specified. If neither CONTAINED nor CONSISTS is specified, an appropriate error message will be printed. If CONTAINED is given, the names used as input must be ELEMENT, GROUP, ENTITY, INPUT and/or OUTPUT names. If the CONSISTS parameter is given the names used as input must be SET, ENTITY, INPUT, OUTPUT and/or GROUP names.

Examples:

```
CM   N=EMPLOYEE-NUMBER   CNTD

CM   FILE=T.DATA   CSTS

CM   CSTS
```

CONTENTS

Purpose: To produce the CONTENTS REPORT.

Prototype: CONTENTS(CONT) [parameter]...

Parameters: FILE(F)[=dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, thatfile is used as theinput file for the command and the report is produced for all the names in the file. When a name is specified by the NAME parameter, the report is produced for that name alone. In any case, the names used as input to the command must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index into the report consisting of an alphabetical listing of all names used in the report and the pages on which they occur.

LEVELS=$\begin{Bmatrix}\text{integer}\\ \text{ALL}\end{Bmatrix}$ Default: LEVELS=ALL

The LEVELS parameter specifies the lowest level of subordinate names to be outputted. The ALL parameter indicates that all subordinate names should be outputted. LEVELS can take on any integer value from 1 to 50.

NCFLAG, NONCFLAG Default: NONCFLAG

The NCFLAG parameter flags all GROUPS in the output reports that do not consist of anything else, and those undefined names which are contained in a GROUP, INPUT, OUTPUT, ENTITY or SET.

Examples: CONTENTS N=VARYING-EMPLOYEE-DATA

CONT F=T.DATA

DATA-BASE-STATISTICS

Purpose: To produce the DATA BASE STATISTICS output.

Prototype: DATA-BASE-STATISTICS(DBS) [parameter]...

Parameters: NAMES, NONAMES Default: NAMES

For each name in the data base, NAMES outputs the number of NUBS with which it is associated.

NAMNUBS, NONAMNUBS Default: NONAMNUBS

NAMNUBS outputs a count of the number of each type of nub, for each name in the data base.

NUBS, NONUBS Default: NUBS

Total number of each type of nub in the data base is part of the output when NUBS is specified.

SYNONYMS(SYN), NOSYNONYMS(NSYN) Default: NOSYNONYMS

SYNONYMS specifies that synonyms should be included in the list of names in the output report. Synonyms do not, however, have nubs associated to them.

Note: This report and the explanation of its parameters is of no direct relevance to the average user of URA. The terminology used here and the value of the output is directed towards those users who have an active part in maintaining the URA system.

Examples: DATA-BASE-STATISTICS

DBS NONUBS

DATA-PROCESS

Purpose: To produce the DATA PROCESS REPORT.

Prototype:

```
DATA-PROCESS(DP) {DATA(D)   } [parameter]...
                 {PROCESS(P)}
```

Parameters: FILE(F)[=dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. The format of the input file must be one name per line.

When a name is given via the NAME parameter, the report is produced only for that name.

DATA(D), PROCESS(P) Default: no default

Since no default exists, one of the above _must_ be specified. If neither DATA nor PROCESS is specified, an appropriate error message will be printed. If DATA is specified, the names used as input to the command must be SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. If PROCESS is specified, the names used as input to the command must be PROCESS names.

Examples:

```
DP   N=PAYROLL-PROCESSING   PROCESS

DP   F=T.DATA  DATA
```

DELETE

Purpose: To delete a name or list of names from the data base. When a name is deleted all of its connections to other names in the data base are also deleted. A permanent record of the change is also generated in the form of the DELETION REPORT.

Prototype:

```
DELETE(DEL) {FILE(F)[=dsname] }
            {NAME(N)=user-name}
```

Parameters: FILE(F)[=dsname], NAME(N)=user-name Default: no default

When the FILE parameter is used and no name is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command and all names in the file are deleted from the data base. The format of the input must be one name per line. When a name is specified by the NAME parameter, that name is deleted from the data base. A warning will be printed if neither of these parameters is specified.

Examples:

```
DELETE   N=FIELD-CHECK-NEW

DEL   FILE=T.DATA
```

DELETE-COMMENT-ENTRY

Purpose: To delete from the data base, for the given name or list of names, those comment entries associated with each comment entry statement specified in the list of parameters. A permanent record of the change is also generated in the form of the DELETED COMMENT ENTRIES report.

Prototype: DELETE-COMMENT-ENTRY(DCOM) $\left\{\begin{array}{l}\text{FILE(F)[=dsname]}\\ \text{NAME(N)=user-name}\end{array}\right\}$ [parameter]...

Parameters: When given as parameters, the comment entries for the following comment entry statement are deleted.

DERIVATION(DER) Default: no comment entries are deleted

DESCRIPTION(DESC)

FALSE-WHILE(FW)

PROCEDURE(PRCD)

TRUE-WHILE(TW)

VOLATILITY(VOL)

VOLATILITY-MEMBER(VOLM)

VOLATILITY-SET(VOLS)

Other parameters

FILE(F)[=dsname], NAME(N)=user-name Default: no default

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, only the specified comment entries for that name are deleted. Either FILE or NAME <u>must</u> be given but not both. The format of the input file must be one name per line.

PRINT, NOPRINT(NP) Default: PRINT

The PRINT parameter initiates the production of a printed DELETED COMMENT ENTRIES report. NOPRINT suppresses the printing.

Examples: DELETE-COMMENT-ENTRY N=NEW-INFO-VALIDATION PRCD DESC

DCOM FILE=T.DATA DESC

DELETE-URL

Purpose: To delete specific URL statements in the user's data base. Those statements used as input to the command are deleted. A permanent record for the change is generated in the form of the DELETED URL output.

Prototype: DELETE-URL(DURL) [parameter] ...

Parameters: INPUT[=dsname] Default: INPUT=*SOURCE*

When INPUT is used and an fdname is specified, the contents of the designated dsname are used as input to the command. This input must be in the same format allowable by the INPUT-URL command (i.e., legal URL statements). The only exception is that no comment entry statements are allowed in the input (DESCRIPTION, for example). The EOF statement terminates input. If no dsname is specified, its value defaults to *SOURCE* so that the URL statements can be entered interactively.

SOURCE(S), NOSOURCE(NS) Default: SOURCE

When the SOURCE parameter is in effect, an as-is source listing (the DELETED URL output) of the deleted URL statements is produced. When the NOSOURCE parameter is given, no source listing is produced.

XREF(X), NOXREF(NX) Default: NOXREF

The user may desire a cross reference for the as-is source listing. This consists of a list of all user-defined names from the input file and the line numbers on which they occur in the DELETED URL outputs. To accomplish this, the user should specify XREF. When NOXREF is in effect, no cross reference is produced.

Example: DELETE-URL INPUT=T.DATA

DURL

DICTIONARY

Purpose: To produce the DICTIONARY REPORT for a name or list of names in the user's data base.

Prototype: DICTIONARY(DICT) [parameter]...

Parameters: FILE(F)[=dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the production of the DICTIONARY REPORT. When a name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

INDEX, NOINDEX Default: NOINDEX

When given, the INDEX parameter specifies the production of an index to the report. This index consists of an alphabetical listing of all names used in the report and the page(s) on which they occur in the report.

NUM-SPACE(NS)=integer Default: NUM-SPACE=2

For ease in reading, the number of lines skipped between dictionary entries can be modified by varying this parameter. NUM-SPACE may take on any value between 0 and 10.

The following parameters specify the information to be included in the DICTIONARY. The "NO" prefix on a parameter specifies that such information not be included for the name(s).

DESCRIPTION(DESC), NODESCRIPTION(NDESC) Default: DESCRIPTION

KEYWORDS(KEY), NOKEYWORDS(NKEY) Default: KEYWORDS

RESPONSIBLE-PD(RPD),
NORESPONSIBLE-PD(NRPD) Default: RESPONSIBLE-PD

SYNONYMS(SYN), NOSYNONYMS(NSYN) Default: SYNONYMS

Examples: DICTIONARY N=PAYROLL-PROCESSING

DICT FILE=T.DATA

ENTITY-IDENTIFIER

Purpose: To produce the IDENTIFIER INFORMATION REPORT

Prototype: ENTITY-IDENTIFIER(EI) $\left\{\begin{matrix}\text{IDENTIFIER(I)}\\ \text{ENTITY(E)}\end{matrix}\right\}$ [parameter]...

Parameters: FILE(F)[=dsname], NAME(N)=user-name   **Default:** FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as in the input file for the command. If a name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

IDENTIFIER(I), ENTITY(E)   **Default:** no default

Since no default is allowed, one of the above must be specified. If neither IDENTIFIER nor ENTITY is specified, an appropriate error message will be printed. If IDENTIFIER is specified, the names used as input to the command must be names used as IDENTIFIERS in the data base. If the ENTITY parameter is given, the names used as input must be defined ENTITY names.

Examples:

```
EI   N=EMPLOYEE-NUMBER   I

EI   FILE=T.DATA   ENTITY
```

FORMATTED-PROBLEM-STATEMENT

Purpose: To produce the FORMATTED PROBLEM STATEMENT for a given name, or list of names and/or to produce this information in the form of PUNCH data.

Prototype: FORMATTED-PROBLEM-STATEMENT(FPS) [parameter]...

Parameters: AMARG(AM)=integer **Default:** AMARG=10

The AMARG parameter indicates the column at which the first name of a name pair is to be outputted. An example of a name pair can be found in the ATTRIBUTE statement where the syntax requires an ATTRIBUTE name and ATTRIBUTE-VALUE.

BMARG(BM)=integer **Default:** BMARG=25

The BMARG parameter indicates the column at which the second name of a name pair is to be outputted.

COMMENT(COM), NOCOMMENT(NCOM) **Default:** COMMENT

The COMMENT option, when in effect, specifies the inclusion of comments for undefined names. The NOCOMMENT option suppresses these comments.

CMARG(CM)=integer **Default:** CMARG=1

The CMARG parameter specifies the number of columns from SMARG the text (comment entry) for a comment entry statement begins.

DEFINE(DEF), NODEFINE(NDEF) **Default:** DEFINE

With the DEFINE option in effect, DEFINE section are included in the report. The NODEFINE option specifies that no DEFINE sections are included in the FORMATTED PROBLEM STATEMENT.

DESG(DG), NODESG(NDG) **Default:** DESG

The DESG option, when in effect, indicates that DESIGNATE sections are provided for SYNONYM names in the FORMATTED PROBLEM STATEMENT. The NODESG option suppresses the production of such output.

FORMATTED-PROBLEM-STATEMENT (cont'd)

EMPTY, NOEMPTY   Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

FILE(F)[=dsname], NAME(N)=user-name   Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for the name specified. The format of the input file must be one name per line.

HMARG(HM)=integer   Default: HMARG=40

This parameter specifies the column where the user defined name in a section header statement are to be printed on the output.

INDEX, NOINDEX   Default: NOINDEX

The INDEX parameter specifies the production of an index for the FPS. This index consists of an alphabetical listing of all user defined names used in the FORMATTED PROBLEM STATEMENT and the page(s) on which they occur.

NEW-LINES(NL), NONEW-LINE(NNL)   Default: NONEW-LINE

When the NEW-LINE parameter is given, the first name of a name list associated with a statement will appear on the line succeeding the statement identifier (name of the statement). The NONEW-LINE parameter initiates the list on the same line as the statement identifier (name of the statement). The NONEW-LINE parameter initiates the list on the same line as the statement identifier.

NEW-PAGE(NPG), NONEW-PAGE(NNPG)   Default: NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the FPS be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

FORMATTED-PROBLEM-STATEMENT (cont'd)

NMARG(NM)=integer   Default: NMARG=20

The NMARG parameter indicates the column in which the name or first name of a name list for any statement will be outputted.

ONE-PER-LINE(OPL), SEVERAL-PER-LINE(SPL)   Default: ONE-PER-LINE

The ONE-PER-LINE option indicates that the names in a name list for any statement will appear on succeeding lines. SEVERAL-PER-LINE option signifies that names in a name list will appear on the same line.

PRINT, NOPRINT(NP)   Default: PRINT

The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the FORMATTED PROBLEM STATEMENT.

PUNCH(P)[=dsname], NOPUNCH   Default: NOPUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no dsname is designated, the data is written into the file, URAFORMLIST. This file is the default PUNCH file for the command. If a dsname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

RNMARG(RM)=integer   Default: RNMARG=70

Specifies the right-hand margin for names in a name list when the SEVERAL-PER-LINE parameter is in effect.

SMARG(SM)=integer   Default: SMARG=5

The SMARG parameter indicates the column in which the statement identifier (name of the statement) will be started.

Examples: FPS N=FIELD-CHECK-NEW

FPS FILE=T.DATA

FREQUENCY

Purpose: **To produce the FREQUENCY REPORT.**

Prototype: **FREQUENCY(FREQ)**

Parameters: **None**

Examples: **FREQ**

HELP

Purpose: To provide the on-line user with a list of possible commands for URA or information about the parameters for a particular URA command.

Prototype: HELP [parameter]...

Parameters: Command-name Default: (see text)

If no command-name is given, a list of currently available URA commands is given. If a command-name is given, then the parameters for that command are presented. Abbreviations for the command-name are also acceptable assuming they are known by the user.

SHORT, LONG Default: SHORT

If SHORT is given, only the parameters for the given command are printed. If LONG is given, explanations of the various parameters are also printed.

Examples:

```
HELP   FPS   LONG

HELP   CONTENTS
```

INPUT-URL

Purpose: To add information to the URA data base to expand or modify the user requirements. A permanent record of the change is also generated in the form of the AS-IS SOURCE LISTING and CROSS REFERENCE.

Prototype: INPUT-URL(IP) [parameter]...

Parameters: DBREF(D), NODBREF(ND) Default: DBREF

The DBREF parameter allows the referencing of the data base by URA in its syntax and semantic analysis. When given, NODBREF allows the analyzer to only perform a syntax check of the input.

INPUT(I)=dsname Default: INPUT=*SOURCE*

When INPUT is specified, the contents of the designated dsname are used as input to the command. This input must be in the format of legal URL statements as specified by ESD TR# 75-88, Vol II "URL USERS MANUAL". The EOF statement terminates input. If the INPUT parameter is not specified, input is read from *SOURCE* so that the URL statements can be entered interactively.

SOURCE(S), NOSOURCE(NS) Default: SOURCE

When the SOURCE parameter is in effect, AS-IS SOURCE LISTING of the input URL is produced. When the NOSOURCE parameter is given, the listing is not produced.

UPDATE(U), NOUPDATE(NU) Default: NOUPDATE

With the UPDATE parameter given, the input will update the URA data base. NOUPDATE indicates that the data base is not to be changed.

XREF(X), NOXREF(NX) Default: NOXREF

XREF specifies that a URA CROSS REFERENCE is to be generated for the AS-IS SOURCE LISTING. This consists of a list of all user defined names from the input file and the line numbers on which they occur.

Examples: INPUT-URL I=IFILE XREF UPDATE

IP U

KWIC

Purpose: To produce a KWIC INDEX for a list of names.

Prototype: KWIC [parameter]...

Parameters: FILE(F)[*dsname] Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. The format of the input file must be one name per line.

DIF=integer Default: DIF=20

DIF is the number of spaces allowed between the keyword and the rest of the name as it appears in the output. In order to fit the output on an 8-1/2 x 11 inch page, the following restriction is necessary:

$$2 \leq DIF \leq 52$$

Examples:

```
KWIC    F=T.DATA

KWIC
```

NAME-GEN

Purpose: To produce the NAME-GEN report and/or retrieve certain names to be put in a PUNCH file and used as input to other commands.

Prototype: NAME-GEN(NG) [parameter]...

Parameters: The following retrieval parameters indicate the types of names to be retrieved and placed in the output file. A "NO" prefix attached to a parameter means that names of that type are not to be retrieved.

Name Type	Default:
ATTRIBUTE(ATTR), NOATTRIBUTE(NATTR)	NOATTRIBUTE
ATTRIBUTE-VALUE(ATTV), NOATTRIBUTE-VALUE(NATTV)	NOATTRIBUTE-VALUE
CONDITION(COND, NOCONDITION(NCOND)	NOCONDITION
ELEMENT(ELE), NOELEMENT(NELE)	NOELEMENT
ENTITY(ENT), NOENTITY(NENT)	NOENTITY
EVENT(EV), NOEVENT(NEV)	NOEVENT
GROUP(GR), NOGROUP(NGR)	NOGROUP
INPUT(INP), NOINPUT(NINP)	NOINPUT
INTERVAL(INT), NOINTERVAL(NINT)	NOINTERVAL
KEYWORD(KEY), NOKEYWORD(NKEY)	NOKEYWORD
MAILBOX(BOX), NOMAILBOX(NBOX)	NOMAILBOX
MEMO, NOMEMO(NMEMO)	NOMEMO
OUTPUT(OUT), NOOUTPUT(NOUT)	NOOUTPUT
PROBLEM-DEFINER(PD), NOPROBLEM-DEFINER(NPD)	NOPROBLEM-DEFINER
PROCESS(PROC), NOPROCESS(NPROC)	NOPROCESS
REAL-WORLD-ENTITY(RWE), NOREAL-WORLD-ENTITY(NRWE)	NOREAL-WORLD-ENTITY
RELATION(RLN), NORELATION(NRLN)	NORELATION
SECURITY(SEC), NOSECURITY(NSEC)	NOSECURITY
SET, NOSET	NOSET
SOURCE(SRC), NOSOURCE(NSRC)	NOSOURCE
SUBSETTING-CRITERION(SSCN), NOSUBSETTING-CRITERION(NSSCN)	NOSUBSETTING-CRITERION
SYNONYMS(SYN), NOSYNONYMS(NSYN)	NOSYNONYMS
SYSTEM-PARAMETER(SYSP), NOSYSTEM-PARAMETER(NSYSP)	NOSYSTEM-PARAMETER
UNDEFINED(UNDF), NOUNDEFINED(NUNDF)	NOUNDEFINED

NAME-GEN (cont'd)

NONE, ALL Default: None

With the NONE parameter, all Name Type switches are set to an "off" position and names of no types (hence, no names) are retrieved. When the ALL parameter is given, all types of names are included in the output except SYNONYMS and UNDEFINED names.

TOTAL Default: no default

The TOTAL parameter specifies the inclusion of all name types including SYNONYMS and UNDEFINED names in the output.

SUBLEVEL(SL)=integer Default: SUBLEVEL=0

This parameter (to be used in conjunction with the SUBPARTS-OF parameter) specifies the level to which the SUBPARTS tree should be traversed and names retrieved. The zero (0) value indicates all levels should be retrieved.

SUBPARTS-OF(SO)=user-name Default: no default

This parameter retrieves all SUBPARTS (down to the level specified by the SUBLEVEL parameter) of the designated user-name. The user-name may be a PROCESS, INPUT, OUTPUT or REAL-WORLD-ENTITY name.

IDENTIFIER(ID),
IDENTIFIER-ELEMENT(IDE), IDENTIFIER-GROUP(IDG)
Default: no default

IDENTIFIER retrieves only those names which are used as IDENTIFIERS in the data base. IDENTIFIER-ELEMENT retrieves only those that are defined as ELEMENT names and are used as IDENTIFIERS, and IDENTIFIER-GROUP retrieves only those defined as GROUP names and are used as IDENTIFIERS.

Specific names can be selected from those names retrieved by the retrieval parameters when using the following selection parameters:

BASIC, NOBASIC Default: BASIC

The BASIC parameter specifies that basic names be included in the output list of the requested name types. "Basic" names are those names which are not SYNONYMS.

NAME-GEN (cont'd)

KEY=user-name   Default: no default

Only those names with the given user-name as a KEYWORD are selected to be part of the output. The user-name must be a name defined as a KEYWORD in the data base.

PD=user-name   Default: no default

Only those names associated with the specified PROBLEM-DEFINER are selected to be part of the output. The user-name must be a PROBLEM-DEFINER name defined in the data base.

Other parameters:

ORDER=$\left\{\begin{matrix}\text{ALPHA}\\\text{BYTYPE}\end{matrix}\right\}$ Default: ORDER=ALPHA

With ORDER equal to ALPHA, the report presents the retrieved names in alphabetical order by name. BYTYPE signifies that the names are grouped by name type with the types alphabetically ordered and names within the same type ordered alphabetically by name.

PRINT, NOPRINT(NP)   Default: PRINT

The PRINT parameter initiates the production of a printed output (NAME GEN); NOPRINT suppresses printing of such a report.

EMPTY, NOEMPTY   Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before the list of names is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

PUNCH(P)[=dsname], NOPUNCH   Default: PUNCH=URANAMES

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and nodsname is designated, the data is written into the file, URANAMES. This file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Examples:

```
NG   PROCESS   RWE
NG   ALL
NG   ALL   KEY=L1
NG   SO=TIME-CARD   PD=WALTER-J-RATAJ   KEY=L1
```

NAME-LIST

Purpose: To produce an alphabetical list of all the names in the data base with their respective name types and any synonyms associated with the name.

Prototype: NAME-LIST(NL) [parameter]....

Parameters: ORDER=$\left\{\begin{array}{l}\text{ALPHA}\\ \text{BYTYPE}\end{array}\right\}$

If ORDER-ALPHA is specified, the list is ordered by the URA name of the object, if ORDER=BYTYPE is specified, the order is alphabetical by name type, with objects of the same type being ordered by name.

Examples:

```
NAME-LIST

NL    ORDER=BYTYPE
```

PICTURE

Purpose: To produce the PICTURE report.

Prototype: PICTURE(PIC) [parameter]...

Parameters: DATA(D), NODATA(ND) Default: DATA

With the DATA parameter in effect, information applicable and available for the given name, or list of names other than structure or flow data is printed on the output. NODATA inhibits such action.

FILE(F)[=dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for that name. In any case, the names used as input to this command must be PROCESS, RWE, SET, ENTITY, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. The format of the input file must be one name per line.

FLOW, NOFLOW Default: FLOW

This parameter presents flow information in the PICTURE report. It presents RECEIVES and GENERATES information between INPUTS and OUTPUTS with PROCESSES and RWEs. It also presents USES and DERIVES information between PROCESSES and data (such as SETS, ENTITIES, GROUPS and ELEMENTS).

INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the PICTURE report. This index consists of all user defined names used in the report, in the report, in alphabetical order and the pages on which they appear in the report.

STRUCTURE(STR), NOSTRUCTURE(NSTR) Default: STRUCTURE

When the STRUCTURE parameter is in effect, the information available in the SUBPARTS, CONSISTS and/or SUBSETS statements and their complementary statements for the input name(s) appears in the report.

Examples:

```
PICTURE   N=PAYCALC-UPDATING

PIC   N=PAYROLL-PROCESSING   NODATA   NOFLOW
```

PRINT-ATTRIBUTE-VALUES

Purpose: To produce the ATTRIBUTE REPORT.

Prototype: PRINT-ATTRIBUTE-VALUES(PAV) [parameter]...

Parameters: FILE(F)[=dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If an dsname is indicated, that file is used as the input file for the command. The input file format must be one ATTRIBUTE name per line. When a name is specified by the NAME parameter, the report is generated for that name only. In any case, only ATTRIBUTE names may be used as input to this command.

Examples:

```
PRINT-ATTRIBUTE-VALUES   FILE=T.DATA

PAV  N=TYPE
```

PROCESS-INPUT-OUTPUT

Purpose: To produce the PROCESS INPUT/OUTPUT report and/or to retrieve data names in the form of PUNCH data to be used as input to other commands.

Prototype: PROCESS-INPUT-OUTPUT(PRIO) [parameter]...

Parameters: FILE(F)=[dsname], NAME(N)=user-name Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. When a single name is specified by the NAME parameter, the report is produced for that name alone. Either FILE or NAME can be used to implement the command but not both. In any case, all the names used as input to this command must be PROCESS names. The input file format is one PROCESS name per line.

DESCRIPTION(DESC), NODESCRIPTION(NDESC)
Default: DESCRIPTION

When the DESCRIPTION parameter is in effect, the comment-entry associated with the DESCRIPTION statement, for each PROCESS names used as input, is retrieved and printed on the report. NODESCRIPTION specifies that this information is not to be retrieved.

PROCEDURE(PRCD), NOPROCEDURE(NPRCD) Default: NOPROCEDURE

When the PROCEDURE parameter is specifed, the comment entry associated with the PROCEDURE statement, for each PROCESS name used as input, is retrieved and printed on the report. With the NOPROCEDURE parameter in effect, this information is not retrieved.

INPUT(INP), NOINPUT(NINP) Default: INPUT

When the INPUT parameter is in effect, all the names of objects used as input to each PROCESS (i.e., names associated with the RECEIVES and USES statements) are retrieved and printed on the report. The NOINPUT parameter specifies that this information is not to be retrieved.

OUTPUT(OUT), NOOUTPUT(NOUT) Default: OUTPUT

When the OUTPUT parameter is in effect, all the names of objects designated as output from each PROCESS (i.e., names associated with the GENERATES, DERIVES, and UPDATES statements) are retrieved and printed on the report. The NOOUTPUT parameter specifies that this information is not to be retrieved.

PUNCH(P)[=dsname], NOPUNCH Default: NOPUNCH

When PUNCH is specified, all those names retrieved as input or output (see the INPUT and OUTPUT parameters) are put into a PUNCH file. When an dsname is given, the list of names is written into that fdname. If no dsname is specified, the list of names is written into URANAMES (which is also the default PUNCH file for NAME-GEN). When NOPUNCH is in effect, the names are not written into a PUNCH file.

EMPTY, NOEMPTY Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before the list of names is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

INDEX, NOINDEX Default: NOINDEX

When given, the INDEX parameter specifies the production of an index into the report. The index consists of all input and output names in the report, in alphabetical order and the page(s) on which they occur in the report.

NEW-PAGE(NPG), NONEW-PAGE(NNPG) Default: NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the PROCESS INPUT/OUTPUT report be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

PRINT, NOPRINT(NP) Default: PRINT

The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the PROCESS INPUT/OUTPUT report.

Examples:

```
PRIO  N=PAYROLL-PROCESSING

PRIO  F=T.DATA  NDESC  NPG  PRCD
```

PUNCH-COMMENT-ENTRY

Purpose: To produce the PUNCHED COMMENT ENTRIES report and/or punch the specified comment entries into a PUNCH file.

Prototype: PUNCH-COMMENT-ENTRY(PCOM) [parameter]...

Parameters: The comment entries associated with the following types of comment entry statements are retrieved when given as parameters.

DERIVATION — Default: no comment entries are retrieved

DESCRIPTION(DESC)

FALSE-WHILE(FW)

PROCEDURE(PRCD)

TRUE-WHILE(TW)

VOLATILITY(VOL)

VOLATILITY-MEMBER(VOLM)

VOLATILITY-SET(VOLS)

Other parameters:

EMPTY, NOEMPTY — Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

FILE(F)[=dsname], NAME(N)=user-name — Default: FILE= URANAMES

When the FILE parameter is used and no dsname is designated, the contents of the file, URANAMES, is used as input to the command. This file is the default PUNCH file for NAME-GEN. If a dsname is indicated, that file is used as the input file for the command. When a name is specified by the NAME parameter, the output is produced for that name alone. The format of tne input file must be one name per line.

PRINT, NOPRINT(NP) — Default: PRINT

The PRINT parameter initiates the production of printed output for the report. When the NOPRINT parameter is given, the PUNCHED COMMENT ENTRIES report is not produced.

PUNCH-COMMENT-ENTRY (cont'd)

PUNCH(P)[=dsname], NOPUNCH   Default: PUNCH= URAPCOM

The PUNCH parameter specifies that PUNCH data should be generated and written into the desiqnated PUNCH file. When the PUNCH parameter is used and no dsname is designated, the data is written into the file, URAPCOM. This file is the default PUNCH file for the command. If a dsname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Examples: PCOM N=PAYROLL-PROCESSING DESC

PCOM F=T.DATA DESC PRCD

RENAME

Purpose: To change the name of some object in the data base and to produce the RENAME REPORT as a permanent record of the change.

```
Prototype:     RENAME(REN)  {OLD(O)=user-name      NEW(N)=user-name}
                            {INPUT(I)=dsname                       }
```

Parameters: INPUT(I)=dsname Default: no default

For multiple name changes, an input file can be used. Each line of the file must consist of the old name followed by the new name. The two names must be separated by one or more blanks.

OLD(O)=user-name Default: no default

The user-name specified here is the name that is to be changed. This name must be defined in the data base.

NEW(N)=user-name Default: no default

The user-name specified here is the name to replace the old name. If the new name is already in the data base, the name will not be changed.

For a single change, both OLD and NEW must be given with legal values.

Examples:

```
RENAME   OLD=EMPLOYEE-CODE   NEW=EMPLOYEE-NUMBER

REN   INPUT=T.DATA
```

REPLACE-COMMENT-ENTRY

Purpose: To replace, for a given name, specific comment entries associated to it. A REPLACED COMMENT ENTRIES report is also printed as a permanent record of the change.

Prototype: REPLACE-COMMENT-ENTRY(RCOM) [parameter]...

Parameters: INPUT(I)=dsname Default: INPUT= URAPCOM

The designated dsname contains the new comment entries that will replace specified old comment entries in the data base. The required format of the file is the same as that punched by PUNCH-COMMENT-ENTRY. If INPUT is not given, the input will be taken from URAPCOM. For each comment entry to be replaced, the following format must be given in the input file:

```
name
comment-entry type
  .
  .
  .
comment entry text
  .
  .
  .          ;
```

Where name is a name defined in the data base, the comment-entry-type (e.g., DESCRIPTION, VOLATILITY, etc.) must be followed by a semicolon. The text following this must also be followed by a semicolon. This sequence of lines can be repeated as many times as necessary in the input file.

PRINT, NOPRINT(NP) Default: PRINT

The PRINT parameter initiates the production of the REPLACED COMMENT ENTRIES report; NOPRINT suppresses printing. The report, if produced, contains both the old and new comment entries.

Examples: RCOM I=PUNCH

SET

Purpose: To set various global switches and parameters.

Prototype: SET (parameter)...

Parameters: DATABASE (DB)= database-name DEFAULT: DATABASE=URADB

The file named is the file in which the database is assembled and stored. This facility allows the user to change databases conveniently and quickly.

ECHO(E)= ON
OFF
Defaults: ECHO=ON in batch mode, ECHO=OFF in conversational mode

With ECHO set equal to ON, the commands are printed on the current output device (*SOURCE*) as they are encountered. This is more desirable in batch (command is printed on line printer) than in on-line mode (you always see the command you type in anyway).

INPUT(I)=dsname Default: INPUT=*SOURCE*

This parameter specifies the file from which subsequent URA commands should be read from.

LINES(L)=integer Default: LINES=45

The number of lines printed by URA per page is set to the indicated number. The default number fits the output to a 8-1/2 x 11 inch page for convenient binding. LINES may take on any value between 10 and 500.

OUTPUT(O)=dsname Default: OUTPUT=*SOURCE*

This parameter specifies where all output resulting from any subsequent URA commands should be written into.

PROMPT(P)= ON
OFF
Defaults: PROMPT=ON in conversational mode, PROMPT=OFF in batch mode.

If PROMPT is set to ON, URA will prompt the user for the correct command or parameters when an error is encountered. With PROMPT set to OFF, URA will ignore invalid commands and parameters and proceed to the next command or parameter.

Examples: SET DB=URADB.DATABASE

SET OUTPUT=OUTPUT.DATA

STOP

Purpose: To terminate execution of the URA and return control to TSO.*

Prototype: STOP

Parameters: NONE

Example: STOP

* This differs from the interrupt as the user cannot issue a EXEC CLI to return control to URA.

STRUCTURE

Purpose: To produce a STRUCTURE report for INPUTS, OUTPUTS, PROCESSES, or REAL-WORLD-ENTITIES.

Prototype: STRUCTURE(STR) [parameter]...

Parameters: INDENT(IND)=integer Default: INDENT=3

The number is the number of spaces to indent each succeeding level in the report. INDENT can take on any value between 1 and 10.

INDEX, NOINDEX Default: NOINDEX

The INDEX parameter, when given, specifies the production of an index to the report, giving the pages on which each undefined name used in the report occurs. NOINDEX specifies that no index should be generated.

URA will produce structure reports for the following name types when given as parameters. (Only one may be given for each report.)

$$\left\{\begin{array}{l}\text{INPUT(INP)}\\ \text{OUTPUT(OUT)}\\ \text{PROCESS(PROC)}\\ \text{REAL-WORLD-ENTITY(RWE)}\end{array}\right\}$$

Default: PROCESS

Examples: STRUCTURE

STR INPUT

SUMMARY

Purpose:	To produce the DATA BASE SUMMARY output.
Prototype:	SUMMARY(SUM)
Parameters:	no parameters
Examples:	SUMMARY SUM

APPENDIX A

Notes on Running URA Under TSO

To create a URA database, the following commands should be given:

```
EXEC NEWDB 'dbname.database'
```

URADB.DATABASE is URA's default name for the URA database. If the user names his database something other an URADB.DATABASE, the name must be specified via the SET command before running any URA commands.

For example, if the user database is called UDB.DATABASE, then the set command must be speficied as:

```
SET DB=UDB.DATABASE
```

The EXEC NEWDB command is only necessary for the initial creation of the database and is not needed subsequently.

To run the analyzer and have the ability to use the URA command language, the following command should be given:

```
EXEC URA
```

URA will come back with a short message and then ask the user to enter a command.

Example URA Runs

An initial run might be:

```
LOGON TSXXXX SIZE(246) PROC(URA)

(enter password when prompted)

EXEC NEWDB DBNAME.DATABASE

EXEC URA

SET DB=DBNAME.DATABASE

INPUT-URL INPUT=INP.DATA XREF UPDATE

NAME-GEN ALL

FPS

STOP

LOGOFF
```

A subsequent run may enter more input and generate some reports:

```
LOGON TSXXXX SIZE(246) PROC(URA)

(enter password when prompted)

EXEC URA

! (attention interrupt)

EDIT OUTPUT.DATA NEW

)blank line RETURN)
```

```
EXEC CLI

SET DB=DBNAME.DATABASE OUTPUT=OUTPUT.DATA

INPUT-URL INPUT=SOME.DATA UPDATE

NAME-GEN PROCESS

PICTURE

NAME-GEN ALL

FPS

STOP

LOGOFF
```

The user could then have the dataset OUTPUT.DATA printed out on a high speed printer to get all the information from the above reports.

APPENDIX B

URA Command Abbreviations

This appendix presents all commands, and parameters to these commands, for URA. It also presents the acceptable abbreviations for these commands and parameters on the right hand side.

Command Name	Parameters	Abbreviations
CHANGE-TYPE		CT
	FILE	F
	NAME	N
	TYPE	T
CONSISTS-COMPARISON		CNC
	FILE	F
CONSISTS-MATRIX		CM
	FILE	F
	NAME	N
	CONTAINED	CNTD
	CONSISTS	CSTS
CONTENTS		CONT
	FILE	F
	NAME	N
	INDEX	
	NOINDEX	
	LEVELS	
	NCFLAG	
	NONCFLAG	
DATA-BASE-STATISTICS		DBS
	NAMES	
	NONAMES	
	NUBS	
	NONUBS	
	NAMNUBS	
	NONAMNUBS	
	SYNONYMS	SYN
	NOSYNONYMS	NSYN
DATA-PROCESS		DP
	FILE	F
	NAME	N
	DATA	D
	PROCESS	P

Command Name	Parameters	Abbreviations
DELETE		DEL
	FILE	F
	NAME	N
DELETE-COMMENT-ENTRY		DCOM
	DERIVATION	DER
	DESCRIPTION	DESC
	FALSE-WHILE	FW
	PROCEDURE	PRCD
	TRUE-WHILE	TW
	VOLATILITY	VOL
	VOLATILITY-MEMBER	VOLM
	VOLATILITY-SET	VOLS
	FILE	F
	NAME	N
	PRINT	
	NOPRINT	NP
DELETE-URL		DURL
	INPUT	I
	SOURCE	S
	NOSOURCE	NS
	XREF	X
	NOXREF	NX
DICTIONARY		DICT
	FILE	F
	NAME	N
	INDEX	
	NOINDEX	
	NUM-SPACE	NS
	DESCRIPTION	DESC
	NODESCRIPTION	NDESC
	KEYWORDS	KEY
	NOKEYWORDS	NKEY
	RESPONSIBLE-PD	RPD
	NORESPONSIBLE-PD	NRPD
	SYNONYMS	SYN
	NOSYNONYMS	NSYN

AD-A041 825 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370--ETC(U)
APR 75 C R MOORE, H J EIDEN
UNCLASSIFIED ESD-TR-75-88-VOL-3 NL

4 OF 5
AD
A 041825

Command Name	Parameters	Abbreviations
ENTITY-IDENTIFIER		EI
	FILE	F
	NAME	N
	IDENTIFIER	I
	ENTITY	E
FORMATTED-PROBLEM-STATEMENT		FPS
	AMARG	AM
	BMARG	BM
	COMMENT	COM
	NOCOMMENT	NCOM
	CMARG	CM
	DEFINE	DEF
	NODEFINE	NDEF
	DESG	DG
	NODESG	NDG
	EMPTY	
	NOEMPTY	
	FILE	F
	NAME	N
	HMARG	HM
	INDEX	
	NOINDEX	
	NEW-LINES	NL
	NONEW-LINES	NNL
	NEW-PAGE	NPG
	NONEW-PAGE	NNPG
	NMARG	NM
	ONE-PER-LINE	OPL
	SEVERAL-PER-LINE	SPL
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	RNMARG	RM
	SMARG	SM

Command Name	Parameters	Abbreviations
FREQUENCY		FREQ
HELP		
	command-name	
	SHORT	
	LONG	
INPUT-URL		IP
	DBREF	D
	NODBREF	ND
	INPUT	I
	SOURCE	S
	NOSOURCE	NS
	UPDATE	U
	NOUPDATE	NU
	XREF	X
	NOXREF	NX
KWIC		
	FILE	F
	DIF	
NAME-GEN		NG
	ATTRIBUTE	ATTR
	NOATTRIBUTE	NATTR
	ATTRIBUTE-VALUE	ATTV
	NOATTRIBUTE-VALUE	NATTV
	CONDITION	COND
	NOCONDITION	NCOND
	ELEMENT	ELE
	NOELEMENT	NELE
	ENTITY	ENT
	NOENTITY	NENT
	EVENT	EV
	NOEVENT	NEV
	GROUP	GR
	NOGROUP	NGR

Command Name	Parameters	Abbreviations
NAME-GEN (cont'd)		
	INPUT	INP
	NOINPUT	NINP
	INTERVAL	INT
	NOINTERVAL	NINT
	KEYWORD	KEY
	NOKEYWORD	NKEY
	MAILBOX	BOX
	NOMAILBOX	NBOX
	MEMO	
	NOMEMO	NMEMO
	OUTPUT	OUT
	NOOUTPUT	NOUT
	PROBLEM-DEFINER	PD
	NOPROBLEM-DEFINER	NPD
	PROCESS	PROC
	NOPROCESS	NPROC
	REAL-WORLD-ENTITY	RWE
	NOREAL-WORLD-ENTITY	NREW
	RELATION	RLN
	NORELATION	NRLN
	SECURITY	SEC
	NOSECURITY	NSEC
	SET	
	NOSET	
	SOURCE	SRC
	NOSOURCE	NSRC
	SUBSETTING-CRITERION	SSCN
	NOSUBSETTING-CRITERION	NSSCN
	SYNONYMS	SYN
	NOSYNONYMS	NSYN
	SYSTEM-PARAMETER	SYSP
	NOSYSTEM-PARAMETER	NSYSP
	UNDEFINED	UNDF
	NOUNDEFINED	NUNDF

Command Name	Parameters	Abbreviations
NAME-GEN (cont'd)		
	BASIC	
	NOBASIC	
	EMPTY	
	NOEMPTY	
	KEY	
	IDENTIFIER	ID
	IDENTIFIER-GROUP	IDG
	IDENTIFIER-ELEMENT	IDE
	NONE	
	ALL	
	ORDER	
	PD	
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	TOTAL	
	SUBLEVEL	SL
	SUBPARTS-OF	SO
NAME-LIST		NL
	ORDER=ALPHA	
	ORDER=BYTYPE	
PICTURE		PIC
	DATA	D
	NODATA	ND
	FILE	F
	NAME	N
	FLOW	
	NOFLOW	
	INDEX	
	NOINDEX	
	STRUCTURE	STR
	NOSTRUCTURE	NSTR

Command Name	Parameters	Abbreviations
PRINT-ATTRIBUTE-VALUES		PAV
	FILE	F
	NAME	N
PROCESS-INPUT-OUTPUT		PRIO
	FILE	F
	NAME	N
	DESCRIPTION	DESC
	NODESCRIPTION	NDESC
	PROCEDURE	PRCD
	NOPROCEDURE	NPRCD
	INPUT	INP
	NOINPUT	NINP
	OUTPUT	OUT
	NOOUTPUT	NOUT
	NEW-PAGE	NPG
	NEW-PAGE	NNPG
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
	EMPTY	
	NOEMPTY	
	INDEX	
	NOINDEX	
PUNCH-COMMENT-ENTRY		PCOM
	DERIVATION	DER
	DESCRIPTION	DESC
	FALSE-WHILE	FW
	PROCEDURE	PRCD
	TRUE-WHILE	TW
	VOLATILITY	VOL
	VOLATILITY-MEMBER	VOLM
	VOLATILITY-SET	VOLS
	EMPTY	
	NOEMPTY	
	FILE	F
	NAME	N

Command Name	Parameters	Abbreviations
PUNCH-COMMENT-ENTRY (cont'd)		
	PRINT	
	NOPRINT	NP
	PUNCH	P
	NOPUNCH	
RENAME		REN
	INPUT	I
	OLD	O
	NEW	N
REPLACE-COMMENT-ENTRY		RCOM
	INPUT	I
	PRINT	
	NOPRINT	NP
SET		
	ECHO	E
	INPUT	I
	LINES	L
	OUTPUT	O
	PROMPT	P
STOP		
STRUCTURE		STR
	INDENT	IND
	INDEX	IND
	NOINDEX	
	INPUT	INP
	OUTPUT	OUT
	PROCESS	PROC
	REAL-WORLD-ENTITY	RWE
SUMMARY		SUM

Glossary

comment entry
The text associated with a comment entry statement. DESCRIPTION, PROCEDURE and VOLATILITY are examples of comment entry statements.

comment entry statement
Any URL statement which allows the contents of the statement to be defined by the user (as is narrative description). The DESCRIPTION and PROCEDURE statements are examples of this.

Control Command
A URA command which passes control information to URA. STOP and SET are Control Commands.

conversational mode
Used synonymously with on-line mode. Refers to interactive use of the computer system through a terminal device.

database
Synonymous with "URA database". This is the information stored and retrieved by URA by means of the Modifier and Report Commands.

data object
Any URA name type that represents some form of data. SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS are all data objects described by URL.

device
Input-output peripheral equipment; for example, a card reader, magnetic tape drive, terminal, etc.

dsname
Any legal dataset name that is to be specified by the user.

ESD TR# 75-88,VolII
"URL User Manual", Version 3.0. This manual specifies and defines all the URL statements processable by URA, their purpose, syntax, etc.

ESD TR# 75-88,VolI
"An Introduction to Computer Aided Requirements Analysis".

header section
Any of the statements allowed in URL that specify the beginning of a set of statements describing the user defined names specified in this statement. See Appendix F of ESD TR# (75-88,) for a complete list of all section types. VolII

input file
Any dataset which contains data to be used by URA commands via INPUT or FILE parameters.

mode
Term used synonymously with "processing mode" which refers to a state where the user has a particular set of operations available to perform required tasks.

Modifier Command
Any URA command which alters the contents of the user's URA database.

name type
Any of the many types of names allowed in URL (i.e., PROCESS, SET, GROUP, etc.). See ESD TR# (75-88, Vol II), "URL Users Manual", Appendix E for a list of all possible name types.

Report Command
Any command available on the URA system whose sole purpose is to retrieve information from the user's URA database.

undefined name
A name that has been entered into the URA database, but has no name type asspciated with it.

URA
The User Requirement Analyzer. Synonym is "Analyzer". Software package which retrieves and inputs information to the URA database.

URA command
Any of the commands that can be used to operate URA. See Appendix I for complete descriptions about each command available on URA.

URA database
Area where URL information is stored (in a coded format) which can then be accessed by the commands allowed by URA.

URA "object"
Synonymous with "name types". Any of the objects that can be defined by URL (a SET, a GROUP, a PROCESS are all objects in URA). Usually referred to as just "object" in this paper.

URL
Acronym for "User Requirement Language" which is the collection of all URL statements allowed for use by URA. See ESD TR# (75-88,Vol II for complete descriptions of all statements available.

URL statements
Those statements specified by ESD TR# (75-88,Vol II. The statement presents one aspect of description for a particular URA "object".

user requirement
A set of requirements specified by users of a proposed system and interpreted by the user into a format acceptable by the organization.

PART III

URA OUTPUTS

Introduction

This part is intended as a guide to understanding the purpose of all outputs generated by the User Requirements Analyzer (URA). It illustrates how each output is to be used and most of all, how it fits into the requirements specification process. It assumed that the reader is familiar with the concepts concerning URL/URA as presented in ESD TR 75-88, Vol I, "Introduction to Computer Aided Rquirements Analysis".

1. OBJECTIVES OF OUTPUTS FOR A URA DATA BASE

There are two major objectives in logical system design:

- To produce a proposed system that is the best possible in terms of what it will cost to build, cost to operate and what it will contribute to the organization.
- To minimize the cost and time to produce this "optimum" target system.

The objective of developing computer-aided methods for use in logical system design is to contribute to the above objectives. At the present time, it is not possible to achieve an optimum solution for both of these objectives. One contribution that can be made by a computer-aided method is to improve the "quality" of the description of the target system. Quality is defined in terms of consistency, unambiguity and completeness.

Consistency means that no statements made in the description contradict, or are incompatible with, other statements and that any particular object is referred to by the same name throughout the description.

Unambiguity means that statements and relationships are made so precisely that interpretation is uniform by all readers.

Completeness means that all necessary relationships are given and no objects have been omitted from the description. The quality objectives can be aided at three levels:

1. URA will enforce consistency and unambiguity through the syntax analysis and reference checks made when data is entered into the URA data base.

2. The URA outputs will make it easier for the user to detect logic errors in the user requirements, unresolved conditions, etc.

3. Some outputs are available to the user to aid in detecting incompleteness and inconsistencies of the user requirements.

The second objective, that of minimizing design cost and time, is aided by transferring much of the clerical workload to the computer. The Analyzer (URA) maintains an up-to-date record of all data collected. The preparation of this data for use by analysts*, management, etc. can be readily retrieved at request. To meet these objectives, URA offers three classes of outputs:

*NOTE: throughout this paper, the terms analyst and user are used synonymously.

- Reports to aid the analyst
- Reports to aid project management
- Final specifications

Outputs to aid the analyst are basically those reports which aid in resolving inconsistencies, ambiguities, and incompleteness in the logical system design user requirements (See Sections 7 7 and 15.)

The outputs of concern to project management pertain to status of the project in the form of amount of information entered into the URA data base, etc. (See Section 16.)

The final specifications are the end result of the logical system design phase using URL/URA. They express all the information in the URA data base in an easy-to-read format. These specifications are intended to be self-contained and not to require any additional reports to explain their contents.

2. TYPES OF URA OUTPUTS

2.1 The URA Command Language*

All data in a URA Data Base is accessed and retrieved via the URA Command Language (see Part II). The Command Language offers three types of commands:

- Report Commands
- Modifier Commands
- Control Commands

Report Commands retrieve data from the URA data base and output this data in some designated format. FORMATTED-PROBLEM-STATEMENT and PICTURE are typical Report Commands.

Data Base Modifier Commands do something to change the contents of the URA data base. The outcome of the modification is always presented by URA in the form of some output report. INPUT-URL is a Modifier Command which generates the URA AS-IS SOURCE LISTING and the URA CROSS REFERENCE, if requested. The contents of the URA AS-IS SOURCE LISTING verifies that the modifications were made by the Analyzer and/or some errors were encountered.

The remaining commands relay control information to URA. SET and STOP are obviously Control Commands. URA does not generate any output for Control Commands unless an error is encountered.

*A complete list of the URA Commands available to the user of URA can be found in Part II .

Though each command can be issued independently it is often advantageous to use some commands in sequence in that output of one command functions as input to another. The most common instance of this is when NAME-GEN (NG) is used to select certain names (say all PROCESSES for example) which are then used as input to a Report Command (possibly PICTURE (PIC), to produce the PICTURE report for all PROCESS names).

2.2 Reports

Inquiries on the data base are implemented in the same manner as reports. An inquiry, in URA, is simply a report for a single name. (Most Report Commands have NAME and FILE parameters for this purpose.) This capability reduces the complexity of learning several different types of commands for different modes of retrieval. Retrieval criteria for any given report can be determined with respect to any combination of the following:

- By Name type (retrieve all PROCESSES, for example)
- By USER (retrieve all designated objects defined by a particular user)
- By KEYWORD (retrieve all designated objects identified by the appropriate KEYWORD)
- By SUBPARTS for a particular object (retrieve all designated information for those names defined as being a SUBPART of a specified object. The object must be a PROCESS, INPUT, OUTPUT or REAL-WORLD-ENTITY).

The retrieval criteria allows the analyst to be completely selective in the contents of the output reports. This means that the user gets no more, no less, than the requested information.

2.3 Special Analysis

Besides the analysis done automatically by URA to check for consistency, unambiguity and completeness, URA offers Problem Analysis reports to check for these qualities after data is stored in the data base. The purpose of these "special analyses" is to perform quality checks on the user requirements, viewing it as the final specifications, to aid the analyst in improving this documentation. (Automatic analysis is done by Modification Commands to perform quality checks on data as it is input into URA.)

2.4 Own Reports

The reports described in the following sections are by no means considered exhaustive of the capabilities of URL/URA. For any particular organization, reports specific to its needs can easily be implemented by the report generation process documented in ISDOS Working Paper No. 100.

3. URA OUTPUT AND ANALYSIS CAPABILITY CLASSIFIED BY LEVEL OF COMPLEXITY

URA provides clerical aids and rapid retrieval capabilities for its users by permitting the data which is collected during the logical system design process to be made available from a data base, thus descreasing, and hopefully eliminating, the need for manually maintained records. Since the information is maintained in a data base, this implies that a user must be able to display this information in various ways to satisfy the needs of the user. The software also provides for the adding of new information, the deleting of unwanted information and the updating of current information about the target system.

The functions for output and analysis of URA may be classified in order of increasing complexity as follows:

Record and Display

Any data gathered by the user and expressed in URL can be entered into the data base and displayed as required. The read process, involves semantic, and syntax checks to be performed on the input data. Diagnostics are given when errors occur in these checks.

Rearrange and Format

Any data in the data base can be rearranged, sorted, and formatted to satisfy external documentation requirements. The FORMATTED USER REQUIREMENTS is an example of this capability. Checks are made that the requested information is available and that the names, for which the retrieval is to be doen, exist in the URA data base.

Checking

URA performs an "in context" check of all data input into the data base. This makes certain that any name in the data base is used consistently throughout the requirements definition. The URA AS-SOURCE LISTING from the INPUT-URL command illustrates this function. A large number of consistency checks are made which all generate error messages when they fail. For example, checks are made so that a name defined as a GROUP is not used as PROCESS, etc.

Analysis

The User Requirements Statement Analyzer will provide more extensive analysis of the relationships that exist between all data and processes than a human analyst can. The sheer size and complexity of today's information processing systems does not permit any analyst or group of analysts to accomplish this task in any amount of time that will not delay the operational status of the target system. The analysis is performed on the user requirements and can lead to simplication of the user requirements through restructuring and grouping, etc. The analysis is carried as far as practical without bringing in physical implementation considerations such as file structures, report formats, hardware characteristics, etc. The matrix-generating reports illustrate this function. This type of analysis generates error messages and warnings when specific "completeness" criteria might fail. For example, a GROUP might be flagged if it did not consist of any GROUPS or LEMENETS. (By definition, a GROUP must consist of other items.)

4. USE OF URA OUTPUTS AND REPORT DESIGN CRITERIA

4.1 Use of URA Outputs

URA outputs are designed to accomplish one or more of the following objectives:

1. Aid the user in communicating with users. The users are defined to be those people who will benefit from the proposed system when finished. As described in ISDOS Working Paper No. 86, the description of the proposed system is based on data collected from users, and must therefore be verified by them. The adequacy of the proposed system must also be verified. Therefore, outputs must be readable by the user and include dictionaries and/or glossaries to aid in keeping terminology consistent.

2. Aid the user in communicating with users.

   - Detecting and correcting inconsistencies, ambiguities, etc., with the data already recorded in the URA data base.

   - Determining what further data is required to make the user requirements "complete."

   - Quick, easy and simple methods for preparing input of new, additional data and corrected data.

   - Provide a ready reference for what has been done.

3. Coordination in the project. These reports are used to aid in coordinating the work of a number of users, for example, by global dictionary and directory facilities. Redundancy of the work effort is avoided by the ability to retrieve current information from the data base to check project status. The dictionary aids in keeping the names used and their respective definitions consistent throughout requirements definition.

4. Produce final specifications. URA must provide the complete user requirements documentation after the users have completed logical design and all information about it is in the data base. A final self-contained set of hard copy documentation is produced for:

   - A permanent reference
   - Final validation for user concurrence
   - Specifications for the physical system design
   - Anyone interested in understanding the system (for those preparing training material, etc.)

   Consequently the documentation must:

   - Contain all information required by organizational standards for such documentation
   - Be an easy-to-read format which can be presented so that anyone familiar with the designed system can get an understanding of it to whatever depth they require.
   - Contain summaries and cross references so that objects can be described at different levels of detail and all information pertaining to one object can easily be found.

5. Aid project management. Reports must be produced to aid project management in reviewing status and evaluating progress of the design process while using URL/URA.

4.2 Design Criteria for an URA Report

In order to serve the uses identified in Section 4.1 effectively, the format of the reports should satisfy the following criteria:

1. Reports should be self-contained and explanatory so that they can be used by a user without consulting a manual. This can be accomplished by dividing reports into three parts:

   Title Section
   Detail Section
   Summary Section (when applicable)

   The Title Section consists of:

   1. Name of Report
   2. Date
   3. Name of Problem (only if set by user)
   4. Parameter values of the command that generated the report

The Detail Section is the main body of the report. It usually displays a selected subset of the contents of the data base in a consistent format. Most of the errors encountered in the processing of the report command have their error diagnostics printed in this section also.

The Summary Section contains report statistics and warning diagnostics. As it is not always practical, the Summary Section is only included in those reports that such information might be useful. The DATA PROCESS report has a Summary Section, for example.

2. The reports should efficiently use the available space, for example, 8-1/2 x 11" pages, so that they can be easily included in the formal documentation. In other words, the reports must conform to the standards of the particular organization's systems department.

5. URA OUTPUT CLASSIFICATION BY CONTENTS AND RELATIONSHIPS ANALYZED

In previous sections, URA outputs have been classified by various criteria. Section 2 classifies outputs by the type of command that generates it (i.e., Modifier or Report), Section 3 classifies outputs on their level of complexity, and Section 4 on the way the outputs are used. Finally, the most important aspect of classifying outputs is by their part in the overall description of the target (or proposed) system.

Many different outputs and reports could be generated from URA, but a large quantity of outputs does not necessarily make up a quality target system. It must be determined what outputs are necessary.

As computer-based information systems become more and more complex, it is impossible to look at the entire system at one time. Therefore, we must look at certain aspects of the system to fit it together. These aspects are the criteria for determining the necessary outputs.

The remaining part of this section will deal with the definition of output classes in the following matter:

1. What aspect of the system description of this class of outputs aids to present.

2. Manual procedures used to obtain this class of outputs.

3. The specific outputs needed in this class and how they are related.

5.1 Input Data and Modification Outputs

1. Throughout the logical system design phase of system building, the description of the target system is constantly under change. New information is added to the description and old information is changed. In an effort to keep track of all these updates to the description, some documentation method must be used.

2. In manual documentation methods, it is commonly the practice to jot down on a scratch pad or verbally communicate to others that a change is being made in the system description. Many times there is no hard copy of the description modification and even if there is, usually there is no standard procedure for doing so. Even when a standard form or standard procedure is specified by the organization, it is difficult to enforce when the procedure for doing so is manual.

3. URA generates separate reports for each different type of modification to the system description and are as follows:

URA AS-IS SOURCE LISTING

URA CROSS REFERENCE

CHANGE-TYPE REPORT

DELETION REPORT

DELETED COMMENT ENTRIES

RENAME REPORT

REPLACED COMMENT ENTRIES

5.2 Complete User Requirements Reference*

1. At any point in the system description, it is often necessary to find out all the relationships that have been specified for a particular object. The reports or outputs to serve this aspect of the system description must present all the information available for any particular object.

2. If this report were to be generated from manually maintained documentation, it would often be very difficult to find all information pertaining to a particular object unless the documentation had adequate referencing facilities. Even if all information can be found, it is usually not in one place. When forms are used for this purpose they usually describe objects such as processes and do not attempt the same for data elements or files, the maintenance of such documentation would be very difficult.

*NOTE: The term "user requirements" is synonymous with "system description" or "description for the proposed system."

3. The information to be presented by this class of outputs can easily be contained in one report, the FORMATTED REQUIREMENTS STATEMENT.

5.3 Directories, Dictionary, Glossaries and KWIC Index

1. The information supplied by directories, dictionaries and glossaries are necessary in coordinating the work effort between several individuals as well as presenting summary information about objects when the complete user requirements reference is too detailed or bulky. These outputs present very selective information about each object.

2. Most manual methods of system description use a list of names to serve the purpose of directories. Unfortunately, there is usually no way to be certain that the list is up-to-date. The same problem is encountered in manually maintaining a data dictionary (or global dictionary, in general). The majority of glossaries used are usually not relevant by the time the system description is finisned. The KWIC index also proves to be a problem to update. In general, directories and glossaries are used on an ad hoc basis in system description where forms are used for the dictionary and KWIC index.

3. The outputs needed to satisfy this aspect of the system description would be in the format of name lists, name indices, and data definition dictionaries. The two outputs:

NAME GEN

URA NAME LIST

allow selective name lists to be generated and formatted as reports. The

URA KWIC INDEX

is a report which is an index into the names in the URA data base based on the keyword identifiers within that name. The two outputs:

DICTIONARY REPORT

PUNCHED COMMENT ENTRIES

provided summary information to serve as dictionaries or glossaries to the user requirements.

5.4 Structure

1. Another aspect of the target system that must be presented is the structure of various objects within the system. For example, as an aid to the physical system designer, it must be known how records relate to files and what data is to be stored within these records. This specific information is necessary in constructing the data structures for the target system.

2. Usually charts are used to present the organizational structure or to illustrate process interrelationships. Narrative description is also used but neither usually follow any standard procedure for doing so. Sometimes structure forms (which present a cross listing of low levels of data into larger groups) are used but this becomes extremely difficult to modify especially when items are contained in several larger groups unless an adequate cross reference listing of this information is available.

3. Of the outputs available for presenting structure within the target system, the following outputs illustrate physical and logical structures specified by the user.

   CONTENTS REPORT

   STRUCTURE

   INTERVAL STRUCTURE

   PICTURE

   PICTURE presents the information available by CONTENTS and STRUCTURE but as a graphical output. Outputs that present data and file relationships and aid to optimize their structures are the following:

   CONSISTS MATRIX

   IDENTIFIER INFORMATION REPORT

   RELATION INFORMATION REPORT

5.5 Flow

1. Another aspect of the system that is required for a complete description is information pertaining to how documents and data are used throughout the system. What is to be used as input at any particular phase of operation, and what are the outputs, are two of the questions to be answered.

2. Flow charts and various diagrams are the most common methods of presenting this information manually. In large systems, this manner of documenting gets extremely bulky (hundreds of pages of flow charts) and even more difficult to update or index. In many cases, the narrative in the rest of the system description does not correspond to the flow charts. Though some computer-aided flow charting packages have been developed they are usually too restrictive to be worth the trouble of implementing them.

3. The flow information presented by these outputs should vary in content by illustrating the relationships of different levels of objects. One may only be interested in the information flow at the data element level and not at the file level. For this reason, the following outputs are deemed necessary to present the aspect of information flow:

PICTURE

DATA PROCESS REPORT

PROCESS INPUT/OUTPUT

5.6 Dynamic Analysis

1. The dynamic analysis aspect of system description is necessary in presenting the manner in which the system changes over time. Questions to be answered here are: how often is the system to perform a certain function, and when? The system will be designed much differently if the objective is to produce a report once a year or six reports a day.

2. Unfortunately, not enough is done in most manual methods of documentation to present this aspect of the system description. Usually any information pertaining to the dynamics of the system are specified (or merely implied) in the narrative descriptions of various objects in the documentation.

3. There are basically two aspects to dynamic analysis. They are finding out: 1) "when" does something happen, and 2) "how often" does it happen. Those outputs with information pertaining to "when" are:

   EVENT/CONDITION REPORT

   DYNAMIC ANALYSIS REPORT

   Those outputs with information pertaining to "how often" are:

   FREQUENCY

   DYNAMIC ANALYSIS REPORT

5.7 Size and Volume

1. Another major aspect of system design that must be taken into consideration is that of system size and volume of processing to be done. This aspect has a tremendous influence on how the system is to be built. The size of the system can make the difference between batch or on-line processing. The volume and size also sets restrictions on cpu and memory size, etc.

2. Again, most of this information is merely contained in the descriptions of various objects in the system description and not located conveniently for analysis.

3. The three major areas that must be described with respect to size are:

system parameters

files

processing

The URA outputs which handle system parameters are:

SYSTEM PARAMETER ANALYSIS

SYSTEM PARAMETER SIZE

which present size information such as range of values, etc. The output:

SET SIZE REPORT

specifies information relating to the size of SETS (or files) in the system. Finally,

PROCESSING VOLUME REQUIREMENTS

presents information about how often a process is initiated and how much information is handled by this process.

5.8 System Properties

1. Aside from the description of individual objects in the user requirements there exists the need of outputs to present various properties of the system concerning cost, implementation and anything else relevant in latter phases of the system building process. This category includes outputs as a result of special analyses related to workload estimation and cost/benefit analysis, etc.

2. There often exists standard format for specifying such information but unfortunately, relatively few standard techniques for deriving this information. As a result, these reports are written in an ad hoc manner, rarely accurate and consequently of little value. Another major problem is encountered when attempting to extract such information from a manually maintained user requirement. Inconsistent or incomplete information leads to an inaccurate system property report.

3. The basic output from which most of the other "special analysis" outputs can be derived is the

   ATTRIBUTE REPORT

   One set of outputs currently being developed to generate a cost/benefit analysis on the present state of the user requirement are the

   COST/EFFECTIVENESS REPORTS

   As various techniques and algorithms are developed to analyze system properties more outputs can be generated.

5.9 Requirements Analysis

1. At any stage in the process of specifying the requirements statement there is a need to check that what has been stated is "correct" and "unambiguous," and what information is needed to make the requirements statement "complete." This is probably one of the most important and most difficult aspects of the logical design process to accomplish. The results of latter phases of the system building process can only be good as the final specification from the logical design phase. By performing sufficient requirements analysis on the requirements statement the final specifications are improved.

2. As most methods of logical system design are done manually, any requirements analysis to be done must be accomplished through the brute force method. By this fact alone, the purpose of performing requirements analysis is many times defeated. It is practically impossible to manually check for inconsistency, incompleteness and ambiguity.

3. Many outputs reflect this capability to chack for consistency and unambiguity of the requirements statement. The AS-IS SOURCE LISTING presents inconsistencies of newly input data before entered into the URA data base. The DATA PROCESS REPORT and the CONSISTS MATRIX present incompleteness information about the requirements statement. The one output designed specifically to present this information, however, is the

   COMPLETENESS/CONSISTENCY REPORT

5.10 Project Management

1. Through the logical design process the requirements definer team must, at various times in the process, generate status reports to project management. These reports hopefully show that the requirements definer (or team) is making progress in their design effort. Quantitative as well as qualitative information pertaining to progress is desirable and should be part of the project status documents.

2. There are many standard forms used for presenting status of the project effort. The status reports can then be compared with a PERT chart or CPM chart to measure progress. The problem encountered in specifying information in the status report is how useful is the work accomplished in approaching completeness of the requirements statement. When all work is done in narrative description it is very difficult to measure progress.

3. The reports generated by URA specifically for project management attempt to present status quantitatively. Quality of the requirements statement can be implied by information from these outputs also. The two present outputs available for project management are:

6. OUTPUT DESCRIPTIONS

For each standard output available by URA, this paper will, in the following sections, describe it based on the following criteria:

1. Name of output
2. Purpose - usage of the output within the logical design process and to whom it benefits (i.e., analyst, management, etc.)
3. Command - name of the URA command and the specific parameters required to initiate this output
4. Options - variations of this report available due to manipulation of the command parameters
5. Contents and Order- what makes up the detail and summary sections of the output and the ordering criteria for this information
6. Analysis - the complexity of the analysis performed by URA to generate this output and the error checking capabilities of the command generating it
7. Examples - copies of the actual printout in order to show the physical features of the output

7. INPUT DATA AND MODIFICATION OUTPUTS

Any data to be entered into the data may be punched on cards or typed in at a terminal, depending on the facilities of the particular installation. In either case, a hard copy record of the input procedure, the URA AS-IS SOURCE LISTING, will be produced. In fact, for any type of modification to the URA data base via the modifier commands specified in Part II , an output report will be generated to given the user a hard copy record of the modification. These outputs can be used by the user to manually check for errors made in the modification procedure.

7.1 URA AS-IS SOURCE LISTING

Purpose: This output is a record of all information input into the URA data base, and is intended as an aid to the analyst. It aids the analyst in finding errors in the input data and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors.

Command: This output can only be produced by the INPUT-URL command with specification of the SOURCE parameter.

Options: There are no options available in the production of this output that effect its content or format.

Contents and Order: This output displays, line for line, the data used as input to the INPUT-URL command. No reordering is done on the input data. Error diagnostics are also printed in this output, usually directly following the line in which the error occurred.

Analysis: URA first performs syntax and semantic checks on each input line before any more complex checking is performed. An "in context" check is made for each name used as input. If the name is not in the user's data base, it is added. If so, a check is made to see that the context in which the name is used in the new input conflicts with the manner in which the name is used in the data base. If there is no conflict, the new relationships stated by the input are entered into the data base. If there is conflict, an error message will be produced and URA will skip to the next input statement.

```
   URA VERSION 740328                                         JUL  7, 1974  18:19.48
                                        ADS-EXAMPLE

                         U R A   A S - I S   S O U R C E   L I S T I N G

PARAMETERS FOR: SYNO

  SOURCE XREF UPDATE DBREF

 LINE  S T M T                                                                   ID FIELD

    1 > /* START OF LEVEL 1      */                                         < >      <
    2 >                                                                     < >      <
    3 >PD: WALTER-J-RATAJ,JOSEPH-ISMITH;                                    < >      <
    4 >     WALTER-J-RATAJ SYN RATAJ;                                       < >      <
    5 >     JOSEPH-ISMITH SYN JI;                                           < >      <
    6 >     BOX: RM-22SH-WEST-ENGINEERING-BLDG ;                            < >      <
    7 >     DESC;                                                           < >      <
    8 >          REQUIREMENTS DEFINERS RESPONSIBLE FOR WRITING THIS         < >      <
    9 >           DESCRIPTION OF THE PAYSYSTEM EXAMPLE.;                    < >      <
   10 >                                                                     < >      <
   11 >INP: WEEKLY-EMPLOYEE-INFORMATION;                                    < >      <
   12 >     DESC;                                                           < >      <
   13 >           THIS INPUT REPRESENTS ALL THE NECESSARY INFORMATION TO    < >      <
   14 >           PRODUCE THE OUTPUTS FROM THE PAYSYSTEM. ;                 < >      <
   15 >     RPD: RATAJ;                                                     < >      <
   16 >     SYN: EMP-INFO;                                                  < >      <
   17 >                                                                     < >      <
   18 >OUT: PAYSYSTEM-OUTPUTS;                                              < >      <
   19 >     DESC;                                                           < >      <
   20 >           THIS OUTPUT REPRESENTS ALL THE REQUIRED OUTPUTS OF THE    < >      <
   21 >           TARGET PAYSYSTEM AS DEFINED BY POLICY. ;                  < >      <
   22 >     RPD: RATAJ;                                                     < >      <
   23 >     SYN: PAYOUTS;                                                   < >      <
   24 >                                                                     < >      <
   25 >SET: PAYROLL-MASTER-INFORMATION;                                     < >      <
   26 >     DESC;                                                           < >      <
   27 >           THIS SET CONTAINS ONE UNIT OF INFORMATION                 < >      <
   28 >           FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,                < >      <
   29 >           THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;            < >      <
   30 >     RPD: RATAJ;                                                     < >      <
   31 >     SYN: PAY-MAST;                                                  < >      <
   32 >                                                                     < >      <
   33 >RWE: DEPARTMENTS-AND-EMPLOYEES;                                      < >      <
   34 >     GENS: EMP-INFO;                                                 < >      <
   35 >     RCVS: PAYOUTS;                                                  < >      <
```

```
(    URA VERSION 740325                                JUL  7, 1974   13:19.48          (
                                 ABS-EXAMPLE (

                    U R A   A S - 1 5   S O U R C E   L I S T I N G

LINE   S T M T                                                              ID FIELD

   36 >       SYN: DEPT-EMP;                                               < >    <
   37 >       DESC;                                                        < >    <
   38 >            THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND < >    <
   39 >        SUPPLY ALL THE INPUTS. ;                                    < >    <
   40 >                                                                    < >    <
   41 >PRC: PAYROLL-PROCESSING;                                            < >    <
   42 >       UPDS PAY-MAST;                                               < >    <
   43 >       RCVS: EMP-INFO;                                              < >    <
   44 >       GENS: PAYOUTS;                                               < >    <
   45 >       DESC;                                                        < >    <
   46 >            THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS       < >    <
   47 >            IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES         < >    <
   48 >            ALL INPUTS AND PRODUCES ALL OUTPUTS.;                   < >    <
   49 >       KEY: LEVEL-1, TARGET-SYSTEM, HIGHEST-LEVEL-PROCESS;          < >    <
   50 >       RPD: RATAJ;                                                  < >    <
   51 >       SYN: PAYPROC;                                                < >    <
   52 >       SOURCE: COMPANY-PROCEDURES-MANUAL;                           < >    <
   53 >                                                                    < >    <
   54 > /* END OF LEVEL 1                                       */         < >    <
   55 >/*  T */    EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF EOF        < >    <
T:       55 LINES    2.232 CPU SEC.    1446. CARDS PER MINUTE
```

7.2 URA CROSS REFERENCE

Purpose: This output is intended as an aid to the analyst in correcting errors that appear in the URA AS-IS SOURCE LISTING and DELETED URL outputs, and also to resolve ambiguities in assigning name types to the undefined names in the LISTING. It is instrumental in correcting errors, as any names that were involved in an error can be quickly referenced to see:

All places in the AS-IS LISTING and DELETED URL where each name is used.

The name type assigned to that name.

From this information, the analyst will be able to determine what information has to be re-entered to correct the error. Since the CROSS REFERENCE also presents all those names which have an ambiguous name type (one that was not defined in previous input), the analyst can resolve these ambiguities by use of the CHANGE-TYPE command or INPUT-URL.

Command: The XREF parameter must be used in the INPUT-URL or DELETE-URL command to get the output.

Options: There are no options available in generating this output.

Contents and Order: This output consists of an alphabetical list of all user defined names, i.e., non-URL names that appear in the AS-IS LISTING or DELETED URL output. For each name that appears in the CROSS REFERENCE, its corresponding name type (as given in the AS-IS LISTING or DELETED output) is printed and a list of all lines in the AS-IS LISTING or DELETED URL output where the name appeared is also given.

Analysis: As the CROSS REFERENCE is basically an index into the AS-IS LISTING or DELETED URL output, nothing much more than rearranging and formatting is done to generate this output aside from some method of maintaining the line number lists for the index.

```
   A VERSION 740328                                         JUL  7, 1974  18:19.48
                                     ADS-EXAMPLE

                               U R A  C R O S S   R E F E R E N C E

 SEQ N A M E                         T Y P E

   1 COMPANY-PROCEDURES-MANUAL       SOURCE
                             52

   2 DEPARTMENTS-AND-EMPLOYEES       REAL-WORLD-ENTITY
                             33

   3 DEPT-EMP                        SYNONYM FOR DEPARTMENTS-AND-EMPLOYEES
                             36

   4 EMP-INFO                        SYNONYM FOR WEEKLY-EMPLOYEE-INFORMATION
                             16       34    43

   5 HIGHEST-LEVEL-PROCESS           KEYWORD
                             49

   6 JI                              SYNONYM FOR JOSEPH-ISMITH
                              5

   7 JOSEPH-ISMITH                   PROBLEM-DEFINER
                              3        5

   8 LEVEL-1                         KEYWORD
                             49

   9 PAY-MAST                        SYNONYM FOR PAYROLL-MASTER-INFORMATION
                             31       42

  10 PAYOUTS                         SYNONYM FOR PAYSYSTEM-OUTPUTS
                             23       35    44

  11 PAYPROC                         SYNONYM FOR PAYROLL-PROCESSING
                             51

  12 PAYROLL-MASTER-INFORMATION      SET
                             25

  13 PAYROLL-PROCESSING              PROCESS
                             41
  JRA VFRSION 740328                                            JUL  7, 1974  18:19.48
                                      ADS-EXAMPLE

                              U R A   C R O S S   R E F E R E N C E

SEQ N A M E                           T Y P E

 14 PAYSYSTEM-OUTPUTS                 OUTPUT
                                  18

 15 RATAJ                             SYNONYM FOR WALTER-J-RATAJ
                                   4    15    22    30    50

 16 RM-228H-WEST-ENGINEERING-BLDG     MAILBOX
                                   6

 17 TARGET-SYSTEM                     KEYWORD
                                  49

 18 WALTER-J-RATAJ                    PROBLEM-DEFINER
                                   3    4

 19 WEEKLY-EMPLOYEE-INFORMATION        INPUT
                                  11
```

7.3 CHANGE-TYPE REPORT

Purpose: As for most of the modifier outputs, this report is also intended to aid the analyst. It presents him with a permanent record of the transactions involved using the CHANGE-TYPE command. This guarantees that any changes via this command will be documented for future reference.

Command: This report is generated any time the CHANGE-TYPE command is issued.

Options: No options for this report.

Contents and Order: This report prints out for each name used as input to the CHANGE-TYPE command, the name, the old name type associated with it and the new name type now assigned to it. Any error diagnostics which may occur during the name type change will also be printed here. The names are printed out in the same order in which tney were read as input to the CHANGE-TYPE command (i.e., FIFO).

Analysis: To ascertain that legal changes are being done, various "checking" facilities must be used to produce this report. For each name type change, URA must check to see that:

i) The name whose name type is to be changed exists in the data base.

ii) The assignment of the new name type is consistent with the context in which the name was used previously.

```
  RA VFRSIUN 740328                                      JUL  7, 1974  18:19.48
                            ADS-EXAMPLE

                               CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

  FILE TYPF=GROUP

 1*  DATE
  OLD TYPF - *** UNKNOWN OR AMBIGUOUS ***
  NEW TYPE - GROUP


 2*  EMPLOYEE-DATA
  OLD TYPE - *** UNKNOWN OR AMBIGUJUS ***
  NEW TYPE - GROUP


 3*  NEW-EMP-VALIDATION
  OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
  NEW TYPE - GROUP


 4*  VALID-T-CARD
  OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
  NEW TYPE - GROUP


 5*  VALID-TERM-INFO
  OLD TYPE - *** UNKNOWN OR AMBIGUOUS ***
  NEW TYPE - GROUP
```

7.4 DELETION REPORT

Purpose: This report serves as a permanent record of all names that have been deleted from the URA data base. It is intended to aid the analyst in keeping track of modifications to the data base. Once there is a record of a particular name being deleted, the analyst has the option of re-using these names.

Command: This report is generated every time the DELETION command is implemented.

Options: There are no options for this report.

Contents and Order: For each name used as input to the DELETION command, it is printed out on the report along with some message concerning the status of the change (i.e., if it did or did not work). The names on the output appear in the same order as read by the DELETION command (i.e., FIFO).

Analysis: The only analysis that must be done to generate this report is to check that the name actually exists in the URA data base before it can be deleted.

```
URA VERSION 740710                                         JUL 11, 1974  21:38.52
                                  ADS-EXAMPLE
                          D E L E T I O N   R E P O R T

PARAMETERS FOR: DELETE

   FILE

DELETED - STUB
DELETED - CHECK
DELETED - FIXED-EMPLOYEE-DATA
```

7.5 DELETED COMMENT ENTRIES

Purpose: Aid to the analyst in serving as a hard copy record for those comment entries deleted from the system description. As stated before, it is desirable to have all modifications to the system description documented.

Command: This output is generated by the DELETE-COMMENT-ENTRY command with the PRINT parameter in effect.

Options: There are no options for this output. No output is generated if the NOPRINT parameter is specified.

Contents and Order: For each comment entry to be deleted from the data base, the following information is printed on the output:

- name in the data base to which the comment entry belonged.
- the type of comment entry (i.e., DESCRIPTION, PROCEDURE, etc.) which is being deleted.
- the full text making up the comment entry.

As in most modifier outputs, the order of the output names with respect to the input names is FIFO.

Analysis: The extent of analysis needed to accomplish the output is basically a rearrangement and formatting of information in the URA data base. Checking is performed to see if the comment entry exists in the data base before it is deleted.

```
     URA VERSION 74032B                                            JUL  7, 1974  18:19.48
                                   ADS-EXAMPLE

                     D E L E T E D   C O M M E N T   E N T R I E S

PARAMETERS FOR: DCOM

   DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
   NOTRUE-WHILE NOFALSE-WHILE PRINT FILE


 1*    TIME-CARD
                DESCRIPTION:
          1             THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN
          2         EMPLOYEE WORKED THE PRECEDING WEEK                                   ;


 2*    PAY-STATEMENT
                DESCRIPTION:
          1             THIS OUTPUT IS THE PAYMENT TO THE EMPLOYEE FOR THE PREVIOUS
          2         WEEKS WORK.                                                          ;


 3*    ERROR-LISTING
                DESCRIPTION:
          1             THIS OUTPUT IS A LISTING OF THAT INPUT DATA THAT FAILED
          2         THE INPUT VALIDATION RULES.                                          ;
```

7.6 RENAME REPORT

Purpose: To aid the analyst in documenting modifications in the system description pertaining to changes in naming objects in the URA data base.

Command: Generated automatically by usage of the RENAME command.

Options: No options available for this report.

Contents and Order: For every name changed by the RENAME command, this report signifies what was the "old name" which appeared in the data base and the "new name" which has taken its place. When the name change is not successful, error diagnostics are also printed specifying the cause of the error. Again, the names are printed on the output in the same order as they are read as input.

Analysis: Some checking must be done before the name change is executed. URA checks that:

- the old name exists in the data base
- the new name is not already used in the data base
- the new name is a legal URL name (see ESD TR #).

If any of these requirements are violated, an error comment will be given.

```
      URA VERSION 740323                                             JUL  6, 1974  16:01.44
                                 ADS-EXAMPLE

                                       RENAME REPORT

PARAMETERS FOR: REN

   INPUT

SEQ OLD NAME                             NEW NAME.
  1 JOSEPH-ISMITH                        HENRY-MILLER
  2 VARYING-EMPLOYEE-DATA                CHANGING-EMPLOYEE-DATA
  3 LEVEL-1                              L1
  4 ERROR-LISTING                        ERROR-LIST
```

7.7 REPLACED COMMENT ENTRIES

Purpose: This output serves as a permanent record of changes to comment entries within the system description. This is intended as an aid to the analyst.

Command: This output is generated by the REPLACE-COMMENT-ENTRY command with the PRINT parameter in effect.

Options: The only option is not having the output printed by specifying the NOPRINT parameter.

Contents and Order: For each "old comment entry" to be replaced, the output prints out, in the following order:

- name to which the "old comment entry" belongs
- the type of comment entry which is being changed
- the entire text of the "old comment entry"
- the entire text of the "new comment entry" which replaces the old one

An error diagnostics referring to problems encountered in executing the command are also printed here.

Analysis: The extent of analysis performed is basically rearrangement and formatting of information in the data base after a check has been made to ascertain that the "old comment entry" exists in the data base and the "new comment entry" is legal for the particular application being used (i.e., not attempting to enter a PROCEDURE comment entry for a SET name).

```
URA VERSION 7+332                                        JUL  6, 1974  16:01.44
                                 ADS-EXAMPLE
                     REPLACED   COMMENT   ENTRIES
PARAMETERS FOR: RCOM

   PRINT


** DELETED COMMENT ENTRY **
  1*    EMPLOYEE
                    DESCRIPTION :
           1                AN EMPLOYEE IS IDENTIFIED BY AN EMPLOYEE NUMBER            ;

** INSERTED COMMENT ENTRY **
  1*    EMPLOYEE
                    DESCRIPTION :
           1              EACH EMPLOYEE IS IDENTIFIED BY A UNIQUE EMPLOYEE NUMBER;


** DELETED COMMENT ENTRY **
  2*    TIME-CARD
                    DESCRIPTION :
           1                THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN
           2            EMPLOYEE WORKED THE PRECEDING WEEK                                ;

** INSERTED COMMENT ENTRY **
  2*    TIME-CARD
                    DESCRIPTION :
           1              THIS INPUT MAINTAINS A HARDCOPY RECORD OF THE HOURS WORKED,
           2              BY EACH EMPLOYEF, IN ANY GIVEN WEEK.;


** DELETED COMMENT ENTRY **
  3*    PAY-STATEMENT
                    DESCRIPTION :
           1                THIS OUTPUT IS THE PAYMENT TO THE EMPLOYEE FOR THE PREVIOUS
           2            WEEKS WORK.                                                       ;

** INSERTED COMMENT ENTRY **
  3*    PAY-STATEMENT
                    DESCRIPTION :
```

```
URA [illegible] 7.132                                    JUL  5, 1974  15:01.44
                         ADS-EXAMPLE

              REPLACED COMMENT ENTRIES

  1           THIS OUTPUT PROVIDES A HARDCOPY RECORD OF EACH EMPLOYEE S
  2           EARNINGS IN ANY GIVEN WEEK.;
```

7.8 DELETED URL

Purpose: This output is a record of all information (besides names and comment entries)* deleted from the data base. It aids the analyst in finding errors in the deletion procedure and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors.

Command: This output can only be produced by the DELETE-URL command with the SOURCE parameter in effect.

Options: There are no options available in the production of this output that effect its contents or format. If the output is not desired, the NOSOURCE parameter may be given.

Contents and Order: This output displays, line for line, the data used as input to the DELETE-URL command. No reordering is done on the input data. Error diagnostics are also printed in this output, usually directly following the line in which the error occurred.

Analysis: URA first performs syntax and semantic checks on each input line before any more complex checking is performed. An "in context" check is made for each name used as input. If the name is not in the user's data base, it is added. If so, a check is made to see that the context in which the name is used in the new input conflicts with the manner in which the name is used in the data base. If there is no conflict, the new relationships stated by the input are entered into the data base. If there is conflict, an error message will be produced and URA will skip to the next input statement.

* The DELETE and DELETE-COMMENT-ENTRY commands perform these functions.

8. COMPLETE USER REQUIREMENTS REFERENCE

This report provides a complete statement of all the data in the URA data base in syntactically correct URL but formatted for ease of reading. This report is not simply a listing of all statements that have been entered into the data base since all the information provided by complementary statements is included. Furthermore, the order is different. This report is intended as a basic reference of the current state of definition of the problem as it provides all cross references.

8.1 FORMATTED PROBLEM STATEMENT

Purpose: This output aids the analyst in specifying a complete system description by generating, for any object(s) used as input to the command, all information currently in the URA data base about the object(s). The analyst can then determine whether the information is incomplete in describing that object or presently invalid. In this sense, the output is an aid to the analyst.

Depending on the particular organization's requirements for final system specifications, this output can make up a large portion of these specifications. Since it presents all the information for any given name, the cross referencing problem in manual methods of documentation is avoided.

Command: This output is generated any time the URA command, FORMATTED-PROBLEM-STATEMENT, is given with the PRINT parameter in effect.

Options: The format of this output can be varied tremendously by reassignment of the format parameters:*

AMARG

BMARG

CMARG

HMARG

NEW-LINES

NEW-PAGE

NMARG

SEVERAL-PER-PAGE

RNMARG

SMARG

Some information in the FPS (FORMATTED-PROBLEM-STATEMENT) can be omitted when using the NOCOMMENT, NODEFINE and NODESG parameters.

An index into the FPS will be produced if the INDEX parameter is specified. This aids in locating names in the FPS output when the output generated is fairly large.

Contents and Order: The output for the FPS is in the same order as the names used as input are read by the command. All the information stored in the data base about that name is printed out in the form of proper URL statements immediately following the name. These URL statements

* See Part II for a detailed explanation of these parameters.

are also ordered, but according to what information they give about the particular name. In general, these statements are ordered in the following manner.

Identification and Description statements

Structure statements

Flow statements

Size and Volume statements

Dynamic information statements

Miscellaneous statements

The Identification and Description statements are statements that are common to all different names such as:

SYNONYMS

DESCRIPTION

SEE-MEMO

KEYWORDS

ATTRIBUTES

The Structure statements are those which specify hierarchical and network relationships such as SUBPARTS, CONSISTS, SUBSETS, UTILIZES, etc.

The Flow statements are those which relate data and processes such as RECEIVES, USES, DERIVED, etc.

The Size and Volume statements correspond to CARDINALITY and VALUES statements, etc.

The Dynamic information statements relates the status of objects over time. Examples of this are the VOLATILITY and HAPPENS statements.

Miscellaneous statements are those contained for the purpose of project management. They are:

RESPONSIBLE-PROBLEM-DEFINER

SECURITY

SOURCE

Analysis: This report basically just takes information from the data base and formats it. A check is made that the names used as input are actually in the data base.

```
  A VERSION 740328                                        JUL  7, 1974  18:19.43
                                ADS-EXAMPLE

                        FORMATTED PROBLEM STATEMENT

PARAMETERS FOR: FPS

   FILE NOINDEX PRINT NOPUNCH SMARG=5 NMARG=20 AMARG=10 BMARG=25 RNMARG=70 CMARG=1 HMARG=40 DESG
   ONE-PER-LINE DEFINE COMMENT NONEW-PAGE NONEW-LINE

   1 DEFINE                                    COMPANY-PROCEDURES-MANUAL
   2      AS A SOURCE:
   3     APPLIES TO:      PAYROLL-PROCESSING;
   4
   5 REAL-WORLD-ENTITY                         DEPARTMENTS-AND-EMPLOYEES;
   6     SYNONYMS ARE:   DEPT-EMP;
   7     DESCRIPTION;
   8            THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND
   9        SUPPLY ALL THE INPUTS.;
  10     GENERATES:      WEEKLY-EMPLOYEE-INFORMATION;
  11     RECEIVES:       PAYSYSTEM-OUTPUTS;
  12
  13 DEFINE                                    HIGHEST-LEVEL-PROCESS
  14      AS A KEYWORD;
  15     APPLIES TO:     PAYROLL-PROCESSING;
  16
  17 PROBLEM-DEFINER                           JOSEPH-ISMITH;
  18     SYNONYMS ARE:   JI;
  19     DESCRIPTION;
  20              PROBLEM DEFINERS RESPONSIBLE FOR WRITING THIS
  21            DESCRIPTION OF THE PAYSYSTEM EXAMPLE.;
  22     MAILBOX:        RM-223H-WEST-ENGINEERING-BLDG;
  23
  24 DEFINE                                    LEVEL-1
  25      AS A KEYWORD;
  26     APPLIES TO:     PAYROLL-PROCESSING;
  27
  28 SET                                       PAYROLL-MASTER-INFORMATION;
  29     SYNONYMS ARE:   PAY-MAST;
  30     DESCRIPTION;
  31            THIS SET CONTAINS ONE UNIT OF INFORMATION
  32            FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,
  33            THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;
  34     UPDATED BY:     PAYROLL-PROCESSING;
  35     RESPONSIBLE-PROBLEM-DEFINER IS:
  36                     WALTER-J-RATAJ;
```

```
                      FORMATTED PROBLEM STATEMENT

PROCESS                                       PAYROLL-PROCESSING;
    SYNONYMS ARE:   PAYPROC;
    DESCRIPTION;
          THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
          IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
          ALL INPUTS AND PRODUCES ALL OUTPUTS.;
    KEYWORDS:       LEVEL-1,
                    TARGET-SYSTEM,
                    HIGHEST-LEVEL-PROCESS;
    RECEIVES:       WEEKLY-EMPLOYEE-INFORMATION;
    GENERATES:      PAYSYSTEM-OUTPUTS;
    UPDATES:        PAYROLL-MASTER-INFORMATION;
    RESPONSIBLE-PROBLEM-DEFINER IS:
                    WALTER-J-RATAJ;
    SOURCE IS:      COMPANY-PROCEDURES-MANUAL;

OUTPUT                                        PAYSYSTEM-OUTPUTS;
    SYNONYMS ARE:   PAYOUTS;
    DESCRIPTION;
          THIS OUTPUT REPRESENTS ALL THE REQUIRED OUTPUTS OF THE
          TARGET PAYSYSTEM AS DEFINED BY POLICY.;
    GENERATED BY:   PAYROLL-PROCESSING;
    RECEIVED BY:    DEPARTMENTS-AND-EMPLOYEES;
    RESPONSIBLE-PROBLEM-DEFINER IS:
                    WALTER-J-RATAJ;

DEFINE                                        RM-228H-WEST-ENGINEERING-BLDG
     AS A MAILBOX:
    APPLIES TO:     WALTER-J-RATAJ,
                    JOSEPH-ISMITH;

DEFINE                                        TARGET-SYSTEM
     AS A KEYWORD:
    APPLIES TO:     PAYROLL-PROCESSING;

PROBLEM-DEFINER                               WALTER-J-RATAJ;
    SYNONYMS ARE:   RATAJ;
    DESCRIPTION:
             PROBLEM DEFINERS RESPONSIBLE FOR WRITING THIS
           DESCRIPTION OF THE PAY SYSTEM EXAMPLE.;
```

```
URA VERSION 740624                                              JUL  7, 1974  13:19.43
                                  ADS-EXAMPLE

                                         FORMATTED PROBLEM STATEMENT

78      MAILBOX:          RM-223H-WEST-ENGINEERING-BLDG;
79      RESPONSIBLE FOR:
80                        PAYROLL-PROCESSING,
81                        PAYROLL-MASTER-INFORMATION,
82                        PAYSYSTEM-OUTPUTS,
83                        WEEKLY-EMPLOYEE-INFORMATION;
84
85  INPUT                                            WEEKLY-EMPLOYEE-INFORMATION;
86      SYNONYMS ARE:   EMP-INFO;
87      DESCRIPTION;
88          THIS INPUT REPRESENTS ALL THE NECESSARY INFORMATION TO
89          PRODUCE THE OUTPUTS FROM THE PAYSYSTEM.;
90      GENERATED BY:   DEPARTMENTS-AND-EMPLOYEES;
91      RECEIVED BY:    PAYROLL-PROCESSING;
92      RESPONSIBLE-PROBLEM-DEFINER IS:
93                      WALTER-J-RATAJ;
94
95  EOF EOF EOF EOF EOF
```

9. DIRECTORIES, DICTIONARY, GLOSSARIES and KWIC INDEX

The directories generated in URA are lists of names. They may list all the names in the data base or can be very selective and only generate those names that satisfy particular criteria such as being PROCESS names with a particular KEYWORD associated with them. The outputs have multiple functions as they can also be used to aid in the production of other outputs. For example, if a list of all PROCESSES were generated and these names put into a file, then the contents of this file could be used as input to another report command, such as PICTURE, to generate PICTURE outputs for all PROCESSES. Of course, these directories aid the analyst in keeping track of names used in the user requirements and can also be used as directories for the final specifications.

The dictionary offers summary information about the names in it and in particular, information pertaining to the identification and narrative description of these names. The dictionary proves to be an aid in maintaining consistency in the user requirements by presenting information about names which can then be compared for similarity and to avoid redundancy. The dictionary could also become part of the final specifications whenever a reference guide to names in the specifications is desirable.

The glossaries provide summary information also, but only in narrative format. The glossaries can serve much of the same purpose as the dictionary except that the glossary is usually a reference into the final specifications, whereas the dictionary can be complete within itself.

9.1 DICTIONARY REPORT

Purpose: This report can be used in several ways. It can be used by the analyst in defining names (avoiding redundancy) to be put in the data base. It can be used as part of the final specifications to serve as summary information to the rest of the documentation. It can also be used as a stand-alone document after the system is completed to aid in defining new objects entered into the system. This report provides a means of communication between user and analyst to assure that requirements are being met and definitions are accurate.

Command: The basic report is automatically generated when the DICTIONARY command is specified.

Options: Certain information can be omitted in the dictionary by specifying any of the parameters:

NODESCRIPTION

NOKEYWORDS

NORESPONSIBLE-PD

NOSYNONYMS

If all these parameters were specified, it would only produce the same information available in the NAME GEN output. Giving these optional parameters allows the user to be more selective in choosing the contents of the DICTIONARY REPORT.

The formatting of the report can be altered somewhat by varying NUM-SPACE, which specifies the number of lines between the dictionary entries.

An index is available into this report when specifying the INDEX parameter. This is beneficial when the dictionary is large.

Contents and Order: As pointed out previously, the contents can vary depending on which parameters are specified. Assuming that all defaults are in effect, for each name used as input to this command, the name and its corresponding name type are printed out along with any information contained in the DESCRIPTION, KEYWORDS, RESPONSIBLE-PROBLEM-DEFINER, and SYNONYMS statements associated with that name. The dictionary entries are ordered in the same way as the list of names used as input to the DICTIONARY command.

```
   RA VERSION 740328                                              JUL  6, 1974  16:01.44
                                  ADS-EXAMPLE

                                DICTIONARY REPORT

PARAMETERS FOR: DICT

   FILE NOINDEX DESCRIPTION SYNONYMS KEYWORDS RESPONSIBLE-PD NUM-SPACE=2


   1 DEPARTMENTS-AND-EMPLOYEES         REAL-WORLD-ENTITY

                  DESCRIPTION:
                        THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND
                     SUPPLY ALL THE INPUTS.

                  SYNONYMS: DEPT-EMP


   2 EMPLOYEE                          REAL-WORLD-ENTITY

                  DESCRIPTION:
                       EACH EMPLOYEE IS IDENTIFIED BY A UNIQUE EMPLOYEE NUMBER

                  SYNONYMS: EMP

                  RESP PD:  WALTER-J-RATAJ


   3 ERROR-LISTING-PRODUCTION          PROCESS

   4 NEW-EMPLOYEE-PROCESSING           PROCESS

   5 PAYROLL-DEPARTMENT                REAL-WORLD-ENTITY

                  DESCRIPTION:
                        THIS DEPARTMENT IS RESPONSIBLE FOR ALL PAYROLL DATA.

                  SYNONYMS: PAY-DEPT

                  RESP PD:  WALTER-J-RATAJ
```

```
  URA VERSION 740328                                     JUL  6, 1974  16:01.44
                          ADS-EXAMPLE

                            DICTIONARY REPORT

6 PAYROLL-PROCESSING      PROCESS

                DESCRIPTION:
                      THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
                      IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
                      ALL INPUTS AND PRODUCES ALL OUTPUTS.

                SYNONYMS: PAYPROC                        PAYROLL

                KEYWORDS: L1                             TARGET-SYSTEM
                      HIGHEST-LEVEL-PROCESS

                RESP PD:  WALTER-J-RATAJ


7 PAYSTATEMENT-PRODUCTION PROCESS

8 TERMINATING-EMP-PROCESSING      PROCESS
```

9.2 URA KWIC INDEX

Purpose: The KWIC INDEX is a tool for the analyst in finding names used in the data base which are logically related by the keywords within those names. The two names W-2-TAX-FORM and TAX-FORM would be related by their common use of the keyword, TAX, and again by the word, FORM. Upon this discovery the analyst may find that both names are being used for the same object (perhaps an inconsistency) or that they are indeed different tax forms and should be better specified (an ambiguity).

Command: The output is generated whenever the KWIC command is given.

Options: A slight modification of the output format is possible through the use of tha DIF parameter. It specifies the number of spaces between the keyword in the name and the rest of the name.

Contents and Order: Every word in front of, or following a "dash" in a URL name is treated as a keyword by the KWIC command. All the names used as input to this command are permuted about the "dashes" in the names. Then, all names and permuted names are sorted and printed on the output, in alphabetical order. The name, BAD-INPUT-DATA, would have three listings in the KWIC output. The first one would be the whole name:

```
BAD-INPUT-DATA
```

The next would be:

```
DATA      BAD-INPUT
```

and the last:

```
INPUT-DATA      BAD
```

Analysis: Aside from the program needed to perform the permuting task on the names,the output is basically produced through rearrangement and formatting of the names used as input to the KWIC command.

```
URA VERSION 740328                                          JUL  6, 1974  16:01.44
                              ADS-EXAMPLE
                          U R A   K W I C   I N D E X

PARAMETERS FOR: KWIC

  DIF=20

  SEQ           N A M E  (PERMUTED)

    1 AND-EMPLOYEES                               DEPARTMENTS
    2 BLDG                          RM-228H-WEST-ENGINEERING
    3 CARD                                               TIME
    4 CHANGING-EMPLOYEE-DATA
    5 COMPANY-PROCEDURES-MANUAL
    6 DATA                                 CHANGING-EMPLOYEE
    7 DEPARTMENT                                     PAYROLL
    8 DEPARTMENTS-AND-EMPLOYEES
    9 EMP-PROCESSING                             TERMINATING
   10 EMPLOYEE
   11 EMPLOYEE-DATA                                 CHANGING
   12 EMPLOYEE-INFO                              TERMINATING
   13 EMPLOYEE-INFORMATION                               NEW
   14 EMPLOYEE-INFORMATION                            WEEKLY
   15 EMPLOYEE-PROCESSING                                NEW
   16 EMPLOYEES                              DEPARTMENTS-AND
   17 ENGINEERING-BLDG                           RM-228H-WEST
   18 ERROR-LIST
   19 ERROR-LISTING-PRODUCTION
   20 HENRY-MILLER
   21 HIGHEST-LEVEL-PROCESS
   22 HIRED-TERMINATED-REPORT
   23 INFO                            TERMINATING-EMPLOYEE
   24 INFORMATION                             NEW-EMPLOYEE
   25 INFORMATION                           PAYROLL-MASTER
   26 INFORMATION                          WEEKLY-EMPLOYEE
   27 J-RATAJ                                       WALTER
   28 LEVEL-PROCESS                                HIGHEST
   29 LIST                                           ERROR
   30 LISTING-PRODUCTION                             ERROR
   31 L1
   32 MANUAL                            COMPANY-PROCEDURES
   33 MASTER-INFORMATION                           PAYROLL
   34 MILLER                                         HENRY
   35 NEW-EMPLOYEE-INFORMATION
   URA VERSION 740328                                              JUL  6, 1974  16:01.44
                                   ADS-EXAMPLE

                                   U R A   K W I C   I N D E X

SEQ          N A M E (PERMUTED)

 36 NEW-EMPLOYEE-PROCESSING
 37 OUTPUTS                                        PAYSYSTEM
 38 PAY-STATEMENT
 39 PAYROLL-DEPARTMENT
 40 PAYROLL-MASTER-INFORMATION
 41 PAYROLL-PROCESSING
 42 PAYSTATEMENT-PRODUCTION
 43 PAYSYSTEM-OUTPUTS
 44 PROCEDURES-MANUAL                                COMPANY
 45 PROCESS                                    HIGHEST-LEVEL
 46 PROCESSING                                       PAYROLL
 47 PROCESSING                                  NEW-EMPLOYEE
 48 PROCESSING                               TERMINATING-EMP
 49 PRODUCTION                                  PAYSTATEMENT
 50 PRODUCTION                                 ERROR-LISTING
 51 RATAJ                                           WALTER-J
 52 REPORT                                  HIRED-TERMINATED
 53 RM-228H-WEST-ENGINEERING-BLDG
 54 STATEMENT                                            PAY
 55 SYSTEM                                            TARGET
 56 TARGET-SYSTEM
 57 TERMINATED-REPORT                                  HIRED
 58 TERMINATING-EMP-PROCESSING
 59 TERMINATING-EMPLOYEE-INFO
 60 TIME-CARD
 61 WALTER-J-RATAJ
 62 WEEKLY-EMPLOYEE-INFORMATION
 63 WEST-ENGINEERING-BLDG                            RM-228H
 64 228H-WEST-ENGINEERING-BLDG                            RM
```

9.3 NAME GEN

Purpose: This output is produced by the most powerful command in the URA system.

1. It is an important aid to the analyst in obtaining other reports and outputs. For example, the analyst can ask for a list of all SET, ENTITY and GROUP names and with this list then ask for a CONTENTS REPORT for those names.

2. It is also used by the analyst as a reference to what names have been used and how they have been used (i.e., if it is a SET rather than an INPUT name).

3. The output can also be used effectively by project management to measure productivity of the project members. This can be done by retrieving a list of all names in the URA defined by a particular user (analyst) and comparing it to previous lists.

4. Finally, the NAME GEN output can become an integral part of the final specifications as it acts as a directory in specifying name lists corresponding to certain selection criteria (a directory of all data elements may be desired before a section which deals with the definition of each element in detail).

Command: Though this output can be varied in so many ways, the most basic form of this output can be the TOTAL parameter. This produces a list of all names in the users URA data base including undefined names and synonyms.

Options: The parameters available in the command allows the user to be very selective in which names he chooses to retrieve from the data base. The user may only choose those names which are of a particular name-type (say GROUP name types) by specifying the GROUP parameter. Only GROUP names will be produced on the output (assuming no other parameters are given). To get all the names associated with more than one name type the user only need specify the parameter that corresponds to eacn of the name types the user requests. In other words, to get a list of all SETS, ENTITIES and GROUPS the appropriate command would be:

```
NAME-GEN      SET     ENTITY     GROUP
```

A list of all names except synonyms and undefined names can be obtained by specifying the ALL parameter.

Other parameters offer even more specific selection criteria. To get a list of all names defined by J-SMITH the command would be:

```
    NAME-GEN        ALL     PD=J-SMITH
```

To get all PROCESSES that had the KEYWORD BATCH associated with them, we could specify:

```
    NAME-GEN       PROCESS      KEY=BATCH
```

The ordering of the output may be changed somewhat through the use of the ORDER parameter. With ORDER=ALPHA, the output is produced with all names in the output in alphabetical order. With ORDER=BYTYPE in effect, all names are ordered first by their corresponding name type (i.e., ATTRIBUTE through UNDEFINED) and then alphabetically by name within name type.

I have presented but a few options for the NAME GEN output. See Part II for the other parameters available with this command.

Contents and Order: The contents of the output vary according to what parameters have been specified. The general format is that of a list of names which are contained in the data base with the name type associated with each name, on the same line, on the right hand side of the output. Again, the ordering of the output depends on how ORDER was assigned. Appropriate messages are generated when no names fit the selection criteria.

Analysis: The command must check each name in the data base to see if they fit the selection criteria specified by the parameters. For each name that it does, it is a simple matter of retrieval and formatting.

```
    URA VERSION 740328                                                 JUL  6, 1974  16:01.44
                                   ADS-EXAMPLE

                                                NAME GEN

PARAMETERS FOR: NG

   ATTRIBUTE ATTRIBUTE-VALUE CONDITION ELEMENT ENTITY EVENT GROUP INPUT INTERVAL KEYWORD MAILBOX
   MEMO OUTPUT PROBLEM-DEFINER PROCESS REAL-WORLD-ENTITY RELATION SECURITY SET SOURCE
   SUBSETTING-CRITERION SYSTEM-PARAMETER NOUNDEFINED NOSYNONYMS BASIC PUNCH PRINT

    1     CHANGING-EMPLOYEE-DATA             ENTITY
    2     COMPANY-PROCEDURES-MANUAL          SOURCE
    3     DEPARTMENTS-AND-EMPLOYEES          REAL-WORLD-ENTITY
    4     EMPLOYEE                           REAL-WORLD-ENTITY
    5     ERROR-LIST                         OUTPUT
    6     ERROR-LISTING-PRODUCTION           PROCESS
    7     HENRY-MILLER                       PROBLEM-DEFINER
    8     HIGHEST-LEVEL-PROCESS              KEYWORD
    9     HIRED-TERMINATED-REPORT            OUTPUT
   10     L1                                 KEYWORD
   11     NEW-EMPLOYEE-INFORMATION           INPUT
   12     NEW-EMPLOYEE-PROCESSING            PROCESS
   13     PAY-STATEMENT                      OUTPUT
   14     PAYROLL-DEPARTMENT                 REAL-WORLD-ENTITY
   15     PAYROLL-MASTER-INFORMATION         SET
   16     PAYROLL-PROCESSING                 PROCESS
   17     PAYSTATEMENT-PRODUCTION            PROCESS
   18     PAYSYSTEM-OUTPUTS                  OUTPUT
   19     RM-228H-WEST-ENGINEERING-BLDG      MAILBOX
   20     TARGET-SYSTEM                      KEYWORD
   21     TERMINATING-EMP-PROCESSING         PROCESS
   22     TERMINATING-EMPLOYEE-INFO          INPUT
   23     TIME-CARD                          INPUT
   24     WALTER-J-RATAJ                     PROBLEM-DEFINER
   25     WEEKLY-EMPLOYEE-INFORMATION        INPUT
```

```
    PSRA VERSION 740328                      (                  JUL  6, 1974  16:52.25        (
                                 ADS-EXAMPLE

                                        NAME GEN

PARAMETERS FOR: NG

     NOATTRIBUTE NOATTRIBUTE-VALUE NOCONDITION NOELEMENT NOENTITY NOEVENT NOGROUP NOINPUT
     NOINTERVAL NOKEYWORD NOMAILBOX NOMEMO NOOUTPUT NOPROBLEM-DEFINER PROCESS NOREAL-WORLD-ENTITY
     NORELATION NOSECURITY NOSET NOSOURCE NOSUBSETTING-CRITERION NOSYSTEM-PARAMETER NOUNDEFINED
     NOSYNONYMS BASIC PUNCH PRINT

      1    ERROR-LISTING-PRODUCTION             PROCESS
      2    FIELD-CHECK-NEW                      PROCESS
      3    FIELD-CHECK-PAYCALC                  PROCESS
      4    FIELD-CHECK-TERM                     PROCESS
      5    FILE-REFERENCING                     PROCESS
      6    NEW-EMPLOYEE-PRINTING                PROCESS
      7    NEW-EMPLOYEE-PROCESSING              PROCESS
      8    NEW-EMPLOYEE-UPDATING                PROCESS
      9    NEW-INFO-VALIDATION                  PROCESS
     10    PAYCALC-INPUT-VALIDATION             PROCESS
     11    PAYCALC-UPDATING                     PROCESS
     12    PAYCHECK-PRINTING                    PROCESS
     13    PAYROLL-PROCESSING                   PROCESS
     14    PAYSTATEMENT-PRODUCTION              PROCESS
     15    TERM-INFO-VALIDATION                 PROCESS
     16    TERMINATING-EMP-PRINTING             PROCESS
     17    TERMINATING-EMP-PROCESSING           PROCESS
     18    TERMINATING-EMP-UPDATING             PROCESS
```

9.4 URA NAME LIST

Purpose: To be used as a directory facility by anyone needing a reference to all names in the user's data base.

Command: The output is generated through use of the NAME-LIST command.

Options: The only option available is the order in which names appear, either alphabetical or alphabetical within type.

Contents and Order: The output is an alphabetical listing of all names in the users data base with each name's corresponding name type followed by synonyms,if any. If there is more than one synonym, each appears on a separate line below the previous one.

Analysis: It is a simple retrieve and format procedure to obtain this output.

```
                              NAME LIST

      NAME                                      TYPE                        SYNONYM

 74   birthdate                                 GROUP                            -
 75   date                                      GROUP                            -
 76   employee-name                             GROUP                            -
 77   employment-date                           GROUP                            -
 78   hourly-job-data                           GROUP                            -
 79   pay-date                                  GROUP                            -
 80   personal-data                             GROUP                            -
 81   salaried-job-data                         GROUP                            -
 82   termination-date                          GROUP                            -
 83   employee-information                      INPUT                       emp-info
 84   employment-termination-form               INPUT                       term info
 85   hourly-employment-form                    INPUT                       h-emp-form
 86   hourly-employmet-form                     INPUT                            -
 87   new-employee-information                  INPUT                       new-info
 88   salaried-employment-form                  INPUT                       s-emp-form
 89   tax-withholding-certificate               INPUT                       tax-cert
 90   tax-withholding-form                      INPUT                            -
 91   time-card                                 INPUT                       t-card
 92   month                                     INTERVAL                         -
 93   week                                      INTERVAL                         -
 94   year                                      INTERVAL                         -
 95   check                                     OUTPUT                           -
 96   error-listing                             OUTPUT                      e-list
 97   hire-report                               OUTPUT                           -
 98   hired-employee-report                     OUTPUT                      hired-report
 99   hourly-employee-report                    OUTPUT                      h-emp-report
100   pay-statement                             OUTPUT                      paycheck
101   pay-stub                                  OUTPUT                           -
102   paysystem-outputs                         OUTPUT                      payouts
103   salaried-employee-report                  OUTPUT                      s-emp-report
104   terminated-employee-report                OUTPUT                      term-report
105   error-documenting-procedure               PROCESS                          -
106   hourly-employee-processing                PROCESS                          -
107   new-employee-processing                   PROCESS                          -
108   paycheck-computation-procedure            PROCESS                          -
109   payroll-processing                        PROCESS                     payproc
110   salaried-employee-processing              PROCESS                          -
111   termination-emp-processing                PROCESS                          -
112   time-card-recovery-procedure              PROCESS                          -
```

```
                              NAME-LIST

     NAME                                    TYPE                                     SYNONYM

113  validity-edit-procedure                 PROCESS                                       -
114  departments-[illegible]-employees       REAL-WORLD-ENTITY                        dept emp
115  employee                                REAL-WORLD-ENTITY                        emp
116  payroll-period                          REAL-WORLD-ENTITY                        pay- edt
117  dept-hourly-emp-relation                RELATION                                      -
118  dept-salaried-emp-relation              RELATION                                      -
119  payroll-master-information              SET                                      pay- agt
120  no-of-departments                       SYSTEM-PARAMETER                              -
121  no-of-hourly-employees                  SYSTEM-PARAMETER                              -
122  no-of-payroll-processing                SYSTEM-PARAMETER                              -
123  no-of-salaried-employees                SYSTEM-PARAMETER                              -
124  one                                     SYSTEM-PARAMETER                              -
125  several                                 SYSTEM-PARAMETER                              -
```

9.5 PUNCHED COMMENT ENTRIES

Purpose: Technically, this output serves as report presenting narrative information in the manner of a glossary. Its main objective, however, is to act as an aid to the analyst in changing comment entries in conjunction with the REPLACE-COMMENT-ENTRY command. The idea of using the output as a glossary (for final specifications perhaps) should not be overlooked, however.

Command: The production of this output is initiated through the PUNCH-COMMENT-ENTRY command with the PRINT parameter in effect. Other parameters must be given in order to specify what comment entries are to be printed.

Options: Any reasonable combination of the different types of comment entries can be specified, as parameters for the output. Care should be taken to assume that the choice of parameters is consistent with the types of names for which the output will be generated (one would not try to get the PROCEDURE comment entries associated with a list of SET names).

Contents and Order: For each name used as input to the command, the name is printed on the output in the order in which it was read (FIFO) and associated with that name the type of comment entry and the text for that comment entry (for each type of comment entry as specified in the parameter list).

Analysis: This is basically a retrieval and format procedure. A message is given when no comment entry is available for a particular comment entry type or the name specified is not in the data base.

```
URA VERSION 740328                                               JUL  6, 1974  16:01.44
                                  ADS-EXAMPLE

                                PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PCOM

  FILE DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
  NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

 1*   EMPLOYEE
              DESCRIPTION;
        1               AN EMPLOYEE IS IDENTIFIED BY AN EMPLOYEE NUMBER ;

 2*   TIME-CARD
              DESCRIPTION;
        1               THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN
        2           EMPLOYEE WORKED THE PRECEDING WEEK ;

 3*   PAY-STATEMENT
              DESCRIPTION;
        1               THIS OUTPUT IS THE PAYMENT TO THE EMPLOYEE FOR THE PREVIOUS
        2           WEEKS WORK. ;
```

10. STRUCTURE OUTPUTS

The structure reports describe the relationships between target system objects as specified by the structure statements of URL which are summarized in Table 10.a. The URL statements only specify one level of structure, such as A is a SUBPART of B, where the report has the ability to present all levels of structure (i.e., A is a SUBPART of B, which is a SUBPART of C, etc.).

Classification of Structure Report

The format of the structure report and the analysis performed to obtain it depend on several structure characteristics:

Class of Structure

Type of Structure

Structure Report Content

Class of Structure

There exists basically three classes of structure in describing IPS structures:

1. Real World Structures: those structures which exist in the organization for which the IPS is to serve. The relationships between departments and personnel (designated by REAL-WORLD-ENTITIES) would be defined in terms of the "real" or actual structure that exists in the organization.

2. Logical Structures: those structures, as specified by the analyst, to be used as an aid in communicating between logical and physical system designers the relationships that exist between data in the system. By specifying logical relationships between SETS, ENTITIES, and ELEMENTS at an early stage, it is easier to design and implement a physical system with corresponding files, records and data elements.

3. Definitional Structures: those structures defined by the user to be used as an aid in the system design process. For example, by structuring PROCESSES into subsequent sub-levels it would be easy to facilitate the "top-down" approach in designing the system.

Type of Structure

Structures may be classified as tree structures (Figure 10.a) or as acyclic networks (Figure 10.b). The difference is that in a tree structure any given node may be related to only one higher level node while acyclic networks allow relationships to exist with several higher level nodes.

Table 10.a One level structure relationships as specified by URL statements. To be read from row to column.

	INPUT	OUTPUT	SET	ENTITY	GROUP	ELEMENT	INTERVAL	RWE	PROCESS
INPUT	SUBPARTS PART		CONTAINED		CONSISTS	CONSISTS			
OUTPUT		SUBPARTS PART	CONTAINED		CONSISTS	CONSISTS			
SET	CONSISTS	CONSISTS	SUBSETS SUBSET	CONSISTS					
ENTITY			CONTAINED		CONSISTS	CONSISTS			
GROUP	CONTAINED	CONTAINED		CONTAINED	CONTAINED CONSISTS	CONSISTS			
ELEMENT	CONTAINED	CONTAINED		CONTAINED	CONTAINED				
INTERVAL							CONSISTS		
RWE								SUBPARTS PART	
PROCESS									SUBPARTS PART

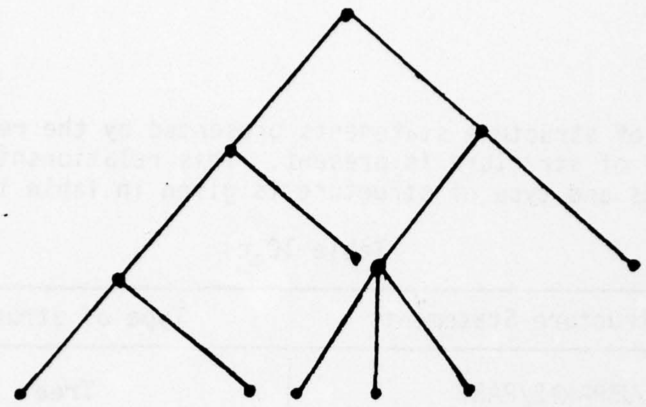


Figure 10.a Tree Structure

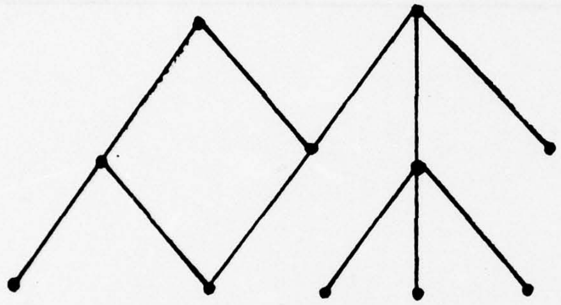


Figure 10.b Acyclic Network

Structure Report Content

There are several objects for which structural relationships can be defined. Each particular type of object can be related to a particular class of structure as shown by Table 10.b.

Table 10.b

objects	statements	class of structure
REAL-WORLD-ENTITY	SUBPARTS/PART	Real World
PROCESS	SUBPARTS/PART	Definitional
INPUT/OUTPUT	SUBPARTS/PART	Definitional
INPUT/OUTPUT	CONTAINED	Definitional
INPUT/OUTPUT	CONSISTS	Logical
INTERVAL	CONTAINED/SETS	Logical
SET	CONSISTS	Logical
SET	SUBSETS/SUBSET	Logical
ENTITY	CONTAINED/CONSISTS	Logical
GROUP	CONTAINED/CONSISTS	Logical
ELEMENT	CONTAINED	Logical

The type of structure statements presented by the report determines what type of structure is present. This relationship between URL statements and type of structure is given in Table 10.c.

Table 10.c

Structure Statements	Type of Structure
SUBPARTS/PART	Tree
SUBSETS/SUBSET	Acyclic Network
CONSISTS/CONTAINED	Acyclic Network

10.1 CONTENTS REPORT

Purpose: This report is intended as an aid to the analyst in communicating with others on the logical structure relationships among data items in the users data base. This report presents the network structure corresponding to the CONSISTS statements associated with SETS, ENTITIES, GROUPS, INPUT and OUTPUTS in the target system description. This report gives the analyst the ability to look at all levels of the structure at one time. This report is very beneficial at the time when some data structure must be devised. This report provides some of the guidelines for doing so.

Command: This report is generated whenever the CONTENTS command is issued.

Options: The report can be generated for one or several names. The user can also be selective in choosing the number of levels to be presented in the structure by assigning the parameter, LEVELS, with a particular value. An index into the report can also be produced when the INDEX parameter is given. Information concerning the incompleteness of the structure can be produced when the NCFLAG parameter is in effect. It flags all those GROUPS used in the report that have not been defined to consist of any subordinate GROUPS and/or ELEMENTS. NCFLAG also flags UNDEFINED names contained in GROUPS, INPUTS, OUTPUTS, ENTITIES or SETS.

Contents and Order: Each name used as input to the command is specified as being a level 1 object. This name is printed along with its corresponding name type next to it in parentheses. All the objects this name CONSISTS of are specified as level 2 objects and printed along with their name type. These level 2 objects then have the objects they consist of printed as level 3 objects, etc. Any system parameter associated with these CONSISTS statements are also included in the output. This procedure is continued to the level which have only ELEMENT names. This is the lowest level.

Analysis: An algorithm is used to traverse the paths in the structure and to specify which objects belong to which levels. This makes it more complicated than a simple retrieval procedure. As this information is retrieved it must also be formatted in an appropriate manner.

```
     .RA VERSION 74032                                 JUL  7, 1974  18:44.24
                              ADS-EXAMPLE

                                   CONTENTS REPORT

PARAMETERS FOR: CONT

   FILE NONCFLAG NOINDEX LEVELS=ALL

   1*     1 NEW-EMPLOYEE-INFORMATION (INPUT)
    1       2   EMPLOYEE-NAME (ELEMENT)
    2       2   NUMBER-OF-DEPENDENTS (ELEMENT)
    3       2   PAYRATE (ELEMENT)
    4       2   EMPLOYEE-NUMBER (ELEMENT)

   2*     1 PAY-STATEMENT (OUTPUT)
    1       2   CHECK (GROUP)
    2       2   STUB (GROUP)
    3        3     EMPLOYEE-NUMBER (ELEMENT)
    4        3     PAYRATE (ELEMENT)
    5        3     GROSS-PAY (ELEMENT)
    6        3     NET-PAY (ELEMENT)

   3*     1 PAYSYSTEM-OUTPUTS (OUTPUT)

   4*     1 TIME-CARD (INPUT)
    1       2   EMPLOYEE-NUMBER (ELEMENT)
    2       2   HOURS-WORKED (ELEMENT)
```

10.2 PICTURE

Purpose: To present flow and structure information in a graphical format. For any given object this output presents its structure by specifying only those objects which are in the level directly above it and those objects in the level directly below it (this is the same structure information that is presented in the FPS). This output is a valuable tool to the analyst for communicating with other people as it is in a format illustrative of the relationships specified for the given object (i.e., all flow information goes from left to right and structure information from top to bottom). (Also described in Section 11.1 on how it relates to flow.)

Command: Structure information is presented when the PICTURE command is used in conjunction with the STRUCTURE parameter. Names used as input to this command must be RWE, PROCESS, INPUT, OUTPUT, SET, ENTITY, GROUP or ELEMENT names.

Options: Flow information is optional in the output and may be omitted by specifying NODATA and NOFLOW. When the OUTPUT is fairly large it may be advantageous to specify the INDEX parameter to generate an index into the output.

Contents and Order: The object being described is contained within a printed box centered in the middle of the page. All objects related to the central object are also contained within boxes arranged around the perimeter of the page. The top line of each box specifies what type of object it is (i.e., PROCESS, ELEMENT, etc.). The bottom line of the boxes specifies what relationship the object within the box has with the central object. For illustrative purposes, lines extend from each box to the central object. The boxes relating to the central base are formatted: five above, five below and six on the right or left. These are maximum allowed to fit on a given page. Any overflow is continued on successive pages.

Analysis: The PICTURE REPORT involves a simple retrieval procedure similar to that in the FPS. The main difference is the manner of presentation. Many more formatting problems are encountered when implementing graphical rather than narrative descriptions for objects.

NOTE: The structure of PICTURE reports is shown on page 81a and the contents of PICTURE reports for each type of object are shown on pages 81b-81h.

```
URA VERSION 74071                                            JUL 11, 1974   21:38.52
                                 ADS-EXAMPLE

                                   PICTURE
PARAMETERS FOR: PIC

   NAME=PAYROLL-PROCESSING NOINDEX NODATA STRUCTURE NOFLOW
```

```
  JRA VERSION 740710                                        JUL 11, 1974   21:38.52
                                     ADS-EXAMPLE

                                PROCESS PICTURE

PAYROLL-PROCESSING




                                   +--PROCESS--+
                                   IPAYROLL-   I
                                   IPROCESSING I
                                   I           I
                                   +-----------+
                                         .
                                         .
                                         .
                                        ...
                                     .   .   .
                                 .       .       .
                              .        . . .        .
                           .         .   .   .         .
                        .          .     .     .          .
                     .           .       .       .           .
                  .           .          .          .           .
               .            .            .            .            .
     +--PROCESS--+  +--PROCESS--+  +--PROCESS--+  +--PROCESS--+  +--PROCESS--+
     IFILE-      I  INEW-       I  ITERMINATIN-I  IPAYSTATEME-I  IERROR-     I
     IREFERENCINGI  IEMPLOYEE-  I  IG-EMP-PROC-I  INT-PRODUCT-I  ILISTING-   I
     I           I  IPROCESSING I  IESSING     I  IION        I  IPRODUCTION I
     +-SUBPARTS--+  +-SUBPARTS--+  +-SUBPARTS--+  +-SUBPARTS--+  +-SUBPARTS--+
```

10.3 STRUCTURE

Purpose: This output is intended as an aid in presenting those structures defined by the analyst for ease of problem specification and Real World Structures. This output presents the structure in a form that the analyst can use in his own work and for communicating with users and system designers. It presents the tree structure corresponding to the SUBPARTS statement associated with REAL-WORLD-ENTITIES, INPUTS, OUTPUTS and PROCESSES. The output gives the user the ability to look at all levels of a particular tree in an easy-to-read format.

Command: The output is generated whenever the STRUCTURE command is given.

Options: For each production of the output, only one type of object (INPUT, OUTPUT, PROCESS or RWE) may be used as input to the command hence, one of these types must be specified as a parameter. An index to the output can also be produced by specifying INDEX as one of the parameters. The format of the output can be modified somewhat by specifying a value for IDENT. This parameter specifies the number of spaces to indent each new level of the structure.

Contents and Order: For any given type of object that the STRUCTURE output is being generated, all names of that type are contained in the output. These names which are not a SUBPART to any other name are specified as level 1 names. Those names which are SUBPARTS to this name are level 2, etc. After all this structure information has been presented a summary section is printed to give certain statistics about the output. For each level of the structure a count is given of the total number of names in that level.

Analysis: An algorithm to traverse the tree structure must be used to print the structure. Also, some simple analysis must be provided to check for looping structures. When a loop is encountered an error message is generated. The formatting procedure for the output varies level by level and involves more than a simple retrieval and format process.

```
  SA VERSION 740328                                               JUL  6, 1974  16:01.44
                                       ADS-EXAMPLE

                                         PROCESS STRUCTURE

 PARAMFTERS FOR: STR

    PROCESS INDENT=3 NOINDEX

COUNT LEVEL NAME

    1     1 PAYROLL-PROCESSING
    2      2   FILE-REFERENCING
    3      2   NEW-EMPLOYEE-PROCESSING
    4       3     NEW-INFO-VALIDATION
    5       3     NEW-EMPLOYEE-UPDATING
    6        4       FIELD-CHECK-NEW
    7       3     NEW-EMPLOYEE-PRINTING
    8      2   TERMINATING-EMP-PROCESSING
    9       3     TERM-INFO-VALIDATION
   10       3     TERMINATING-EMP-UPDATING
   11        4       FIELD-CHECK-TERM
   12       3     TERMINATING-EMP-PRINTING
   13      2   PAYSTATEMENT-PRODUCTION
   14       3     PAYCALC-INPUT-VALIDATION
   15       3     PAYCALC-UPDATING
   16        4       FIELD-CHECK-PAYCALC
   17       3     PAYCHECK-PRINTING
   18      2   ERROR-LISTING-PRODUCTION



LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT    LEVEL COUNT
    1     1        2     5        3     9        4     3
```

10.4 CONSISTS MATRIX

Purpose: Presents the information available by the CONTENTS REPORT but in a more analyzable format. Through the use of a matrix this output proves valuable in illustrating relationships between objects which may not be apparent when looking at the CONTENTS REPORT. For example, through matrix representation of structure it is easy to see when two or more objects may consist of the same or very similar information. This can then be used to improve the logical description as well as an aid when designing the physical system. This output is intended as an aid primarily to analyst and physical system designers.

Command: This output is generated whenever the CONSISTS MATRIX command is given in conjunction with either the CONSISTS or CONTAINED parameter.

Options: When the CONTAINED parameter is used, all the names used as input to the command are designated a number to represent one of the columns in the matrix. All the objects that are CONTAINED in the input names are designated numbers corresponding to a particular matrix row. A CONTAINED relationship between column and row is specified by an asterisk (*). When the CONSISTS parameter is used, all the names used as input to the command are designated a number to represent a particular row in the matrix. All the objects that CONSIST of these input names are designated numbers corresponding to a particular column. A CONSISTS relationships between row and column is specified by an asterisk (*).

Contents and Order: The first part of the output signifies which names have no relationships specified in the matrix. Then a listing is presented to show correspondence of a particular name to its row number and/or column number in the matrix. The assignment of numbers is dependent on the order read in by the command. Next comes the matrix itself with asterisks designating the CONSISTS/CONTAINED relationships between row numbers and column numbers. Objects represented by rows are CONTAINED in the objects represented by the columns. A blank row designates that the object represented by the row is not CONTAINED by any other object. A blank column designates that the object represented by the column does not CONSIST of any objects. Finally, summary information is printed to present some statistics on the matrix information. This summary presents, for each name represented by a row, the number of columns to which it is related (the number of objects that CONTAIN it) and, for each name represented by a column, the number of objects it CONSISTS OF.

Analysis: A check is done to see which names used as input have nc relationships specified in the matrix. An algorithm for presenting data base information in a matrix format must be read and statistical information must be extracted for presentation in the summary section of the output.

```
                                                          JUL 11, 1974   21:38.52
                                     ADS-EXA
                               CONSISTS MATRIX REPORT

PARAMETERS FOR: CA

  FILE CONSISTS

  PSA315:CONCOL: THE FOLLOWING DO NOT CONSIST OF ANYTHING:

        BAD-INPUT-DATA
        DATE
        EMPLOYEE-DATA
        EXPLICIT-PERSONAL-DATA
        NEW-EMP-VALIDATION
        PAY-UNITS
        RELATED-PERSONAL-DATA
        TAX-UNITS
        VALID-T-CARD
        VALID-TERM-INFO

    **ROW**                                         **COLUMN**

  1 EMPLOYEE-NAME                                  1 BAD-INPUT-DATA
    (ELEMENT)                                        (GROUP)
  2 NUMBER-OF-DEPENDENTS                           2 DATE
    (ELEMENT)                                        (GROUP)
  3 PAYRATE                                        3 EMPLOYEE-DATA
    (ELEMENT)                                        (GROUP)
  4 EMPLOYEE-NUMBER                                4 EXPLICIT-PERSONAL-DATA
    (ELEMENT)                                        (GROUP)
  5 VARYING-EMPLOYEE-DATA                          5 NEW-EMP-VALIDATION
    (ENTITY)                                         (GROUP)
  6 TERMINATION-CODE                               6 NEW-EMPLOYEE-PART
    (ELEMENT)                                        (GROUP)
  7 NEW-EMPLOYEE-PART                              7 PAY-UNITS
    (GROUP)                                          (SET)
  8 TAX-RATE                                       8 PAYROLL-MASTER-INFORMATION
    (ELEMENT)                                        (SET)
  9 YTD-DEDUCT                                     9 RELATED-PERSONAL-DATA
    (ELEMENT)                                        (GROUP)
 10 YTD-GROSS                                     10 TAX-UNITS
    (ELEMENT)                                        (SET)
                                                  11 TERMINATED-EMPLOYEE-PART
```

```
JRA VERSION 740710                                           JUL 11, 1974   21:38.52
                                  ADS-EXAMPLE

                             CONSISTS MATRIX REPORT

**THE NUMBER OF COLUMNS THAT CONTAIN THE ROWS**

    ROW                                TYPE                       COUNT

 1  EMPLOYEE-NAME                      ELEMENT                        2

 2  NUMBER-OF-DEPENDENTS               ELEMENT                        2

 3  PAYRATE                            ELEMENT                        2

 4  EMPLOYEE-NUMBER                    ELEMENT                        2

 5  VARYING-EMPLOYEE-DATA              ENTITY                         1

 6  TERMINATION-CODE                   ELEMENT                        1

 7  NEW-EMPLOYEE-PART                  GROUP                          1

 8  TAX-RATE                           ELEMENT                        1

 9  YTD-DEDUCT                         ELEMENT                        1

10  YTD-GROSS                          ELEMENT                        1

**THE NUMBER OF ROWS CONTAINED IN THE COLUMNS**

    COLUMN                             TYPE                       COUNT

15  VARYING-EMPLOYEE-DATA              ENTITY                         5

 6  NEW-EMPLOYEE-PART                  GROUP                          4

11  TERMINATED-EMPLOYEE-PART           GROUP                          3

 8  PAYROLL-MASTER-INFORMATION         SET                            1

12  VALID-NEW-INFO                     GROUP                          1
```

```
                                                   JUL 11, 1974  21:53.57

                                  [illegible] MATRIX REPORT

**THE NUMBER OF ITEMS CONTAINED IN THE ROWS**

     ROW                                TYPE                   COUNT

 1   EMPLOYEE-NAME                      ELEMENT                  2

 2   NUMBER-OF-DEPENDENTS               ELEMENT                  2

 3   PAY-RATE                           ELEMENT                  2

 4   EMPLOYEE-NUMBER                    ELEMENT                  2

 5   VARYING-EMPLOYEE-DATA              ENTITY                   1

 6   TERMINATION-CODE                   ELEMENT                  1

 7   NEW-EMPLOYEE-PART                  GROUP                    1

 8   TAX-RATE                           ELEMENT                  1

 9   YTD-DEDUCT                         ELEMENT                  1

10   YTD-GROSS                          ELEMENT                  1

**THE NUMBER OF ROWS CONTAINED IN THE COLUMNS**

     COLUMN                             TYPE                   COUNT

15   VARYING-EMPLOYEE-DATA              ENTITY                   5

 6   NEW-EMPLOYEE-PART                  GROUP                    4

11   TERMINATED-EMPLOYEE-PART           GROUP                    3

 3   PAYROLL-MASTER-INFORMATION         SET                      1

12   VALID-[illegible]                  GROUP                    1
```

BEST AVAILABLE COPY

```
     VERSION 740719                                                 JUL 11, 1974   21:38.52
                                    ADS-EXAMPLE

                                CONSISTS MATRIX REPORT

  1  BAD-INPUT-DATA                GROUP                                     0

  2  DATE                          GROUP                                     0

  3  EMPLOYEE-DATA                 GROUP                                     0

  4  EXPLICIT-PERSONAL-DATA        GROUP                                     0

  5  NEW-EMP-VALIDATION            GROUP                                     0

  7  PAY-UNITS                     SET                                       0

  9  RELATED-PERSONAL-DATA         GROUP                                     0

 10  TAX-UNITS                     SET                                       0

 13  VALID-T-CARD                  GROUP                                     0

 14  VALID-TERM-INFO               GROUP                                     0
```

10.4a CONSISTS COMPARISON REPORT

Purpose: Derives information available from the CONSISTS statements. For each input object, its lowest level constituents (the bottom nodes of the CONSISTS network structure) are retrieved. A matrix illustrates the relationships between input names and lowest level constituents. From this, the user can determine the degree of similarity between data (number of low level constituents in common between any two data objects) and redundancy (two objects which consist of the same low level constituents). This is intended as an aid to the analyst to improve the quality of the problem statement.

Command: This report is generated whenever the CONSISTS-COMPARISON command is issued.

Options: No options are currently available to change format or content of the report.

Contents and Order: The first part of the output specifies which names have no relationships designated in the matrices. Then a listing is presented to show the correspondence of input names to a particular row and of derived low level constituents to a particular column for the BASIC CONTENTS MATRIX. Next comes the matrix itself with an asterisk designating that a data object, represented by a row, CONSISTS (directly or indirectly) of a low level data object, represented by a column. The CONTENTS SIMILARITY MATRIX presents an analysis performed on the first matrix. All the names used as input are represented by a column number and row number in the matrix. (The numbers are the same so that any given object is represented by row J and column J.) The matrix should be read from row to column as saying: the data object represented by row I has an integer number of low level constituents in common with the data object represented in column J. When I=J, the number of low level constituents of any object in common with itself is presented. This is, of course, the total number of constituents for that given data object. The final section of this report is the CONTENTS SIMILARITY ANALYSIS which presents those input names that have identical lowest level constituents or which are strict subsets (at the lowest level) of other input names.

Analysis: A check is done to see that names used as input have no relationships specified in the matrices. Also, an algorithm must be used to present the data used as input and data retrieved into a matrix format. Statistical information is derived for presentation in the second matrix. Finally, analysis is done for similarity of the constituents for any two or more input names.

10.5 IDENTIFIER INFORMATION REPORT

Purpose: Relates IDENTIFIER names to the ENTITIES that they identify to aid in improving the logical structure specified by the user requirement and also as a tool to the physical system designer in determining the logical and/or physical data structures. A matrix is produced to illustrate these relationships.

Command: This report is generated through usage of the ENTITY-IDENTIFIER command used in conjunction with either the IDENTIFIER or ENTITY parameter.

Options: When the IDENTIFIER parameter is used all names used as input to the command must be IDENTIFIER names. Each of these IDENTIFIER names corresponds to a single row of the matrix and those ENTITIES which they identify correspond to a particular column of the matrix.

When the ENTITY parameter is used all names used as input to the command must be ENTITY names. Each of these ENTITY names corresponds to a single column of the matrix and the IDENTIFIERS that identify these ENTITIES correspond to a particular row number of the matrix.

Contents and Order: The first part of the output signifies which names have no relationships specified in the matrix. Then a listing is presented to show the correspondence of a particular name to its row number or column number in the matrix. The assignment of numbers is dependent on the order read in by the command. Next comes the matrix itself with asterisks designating the IDENTIFIES relationship between row and column numbers. The IDENTIFIERS represented by rows IDENTIFY the ENTITIES represented by the columns. A blank row designates that the IDENTIFIER represented by the row does not IDENTIFY any ENTITIES. A blank column designates that the ENTITY represented by the column is not IDENTIFIED by any IDENTIFIERS. Finally, summary information is printed to present some statistics on the matrix information. This summary presents, for each IDENTIFIER represented by a row, the number of ENTITIES it IDENTIFIES and, for each ENTITY represented by a column the number of IDENTIFIERS that IDENTIFY it.

Analysis: A check is done to see which names used as input have no relationships specified in the matrix. Also, an algorithm for presenting data base information in a matrix format must be used. Statistical information must also be extracted for presentation in the summary section of this output.

```
                                                        JUL 11, 1974  16:18.20
                                   AUS-EX   LE

                              IDENTIFIER INFORMATION REPORT

PARAMETERS FOR: FI

  FILE ENTITY

    **ROW**                            **COLUMN**

1 EMPLOYEE-NUMBER                    1 FIXED-EMPLOYEE-DATA
  (ELEMENT)                            (ENTITY)
                                     2 VARYING-EMPLOYEE-DATA
                                       (ENTITY)

                          THE ROWS ARE IDENTIFIERS OF THE COLUMNS WITH *S

                                           12
                                          +--+
                                        1 I**I
                                          +--+
```

10.6 RELATION INFORMATION REPORT

Purpose: To present information about RELATIONS to aid in the system description and eventually the physical system design. This output aids in detecting redundancies, incompleteness, and inconsistencies in the specification of the RELATION information in the user requirement This output is thus intended as a tool for the analyst and the physical system designer.

Command: This output is not currently available on the URL/URA system.

Contents and Order: Narrative information is intended to be part of the output to function simply as RELATION definition. Three matrices provide more analyzable information about the RELATIONS. The first matrix describes the mapping, and degree of mapping (e.g., one to many) between ENTITIES. The second matrix describes the interaction between the RELATIONS and corresponding ENTITIES. The third matrix provides information about the RELATIONS and corresponding ASSOCIATED DATA. Summary information is provided to aid in resolving incompleteness and/or inconsistencies in the RELATION descriptions. RELATIONS will be grouped based on their utilization, type of usage and parent-child relationships.

10.7 INTERVAL STRUCTURE

Purpose: This output presents the structure defined for time intervals within the system description. For definitional purposes it is important to specify how the organization perceives time. What time intervals are relevant, what are they called and how they interact with each other are a few of the questions to be answered by this output. This output presents the information available from the CONSISTS statement for INTERVALS.

Command: This output is not currently available on the URL/URA system.

Contents and Order: The structures for INTERVALS differ from those for data in the sense that an INTERVAL structure, once defined, remains static whereas the number of occurrences in a data structure varies over time. The output must therefore take into account the system parameters used in conjunction with the CONSISTS statements in the INTERVAL Section.

11. FLOW ANALYSIS OUTPUTS

Flow analysis can be (and has been) presented in many different formats, from narrative description to flow charting. Still, each method of documenting this aspect of the system description has its advantages and disadvantages and none of them seem sufficient on their own. URA allows the users to present the system flow in three formats:

Graphical flow chart (PICTURE)

Matrix representation (DATA PROCESS REPORT)

Narrative description (PROCESS INPUT/OUTPUT)

The purpose of different formats is to allow the same (or similar) information to be presented for different applications (i.e., for communication between users or as a tool for the analyst in improving the flow network).

The flow information is derived from the RECEIVES, GENERATES, USES, DERIVES and UPDATES statements in URL.

11.1 PICTURE

Purpose: The PICTURE output allows the flow information (through RECEIVES, DERIVES, etc.) to be presented in a graphical or flow chart format. This proves to be a very easy way to look at the system though it is only possible to see a small part of the system at any one time. The PICTURE is very useful in communicating a view of the system to others. PICTURES can also play an important part in the final specifications as it presents a graphical summary of the narrative description for any given object. (The PICTURE output is also described in Section 10.2 on how it presents structure information.)

Command: Flow information is presented when the PICTURE command is used in conjunction with the FLOW and DATA parameters. Names used as input to this command must be RWE, PROCESS, INPUT, OUTPUT, SET, ENTITY, GROUP or ELEMENT names.

Options: Structure information is optional in the output and may be omitted by specifying NOSTRUCTURE. When the output to this command gets to be fairly large it may be advantageous to specify the INDEX parameter to generate an index into the output.

Contents and Order: The object being described is contained within a printed box centered in the middle of the page. All objects related to the central object are also contained within boxes arranged around the perimeter of the page. The top line of each box specifies what type of object it is (i.e., PROCESS, ELEMENT, etc.). The bottom line of the boxes specifies what relationship the object within the box has with the central object. For illustrative purposes, lines extend from each box to the central object. The boxes relating to the central bases are formatted: five above, five below and six on the right or left. These are maximum allowed to fit on a given page. Any overflow is continued on successive pages.

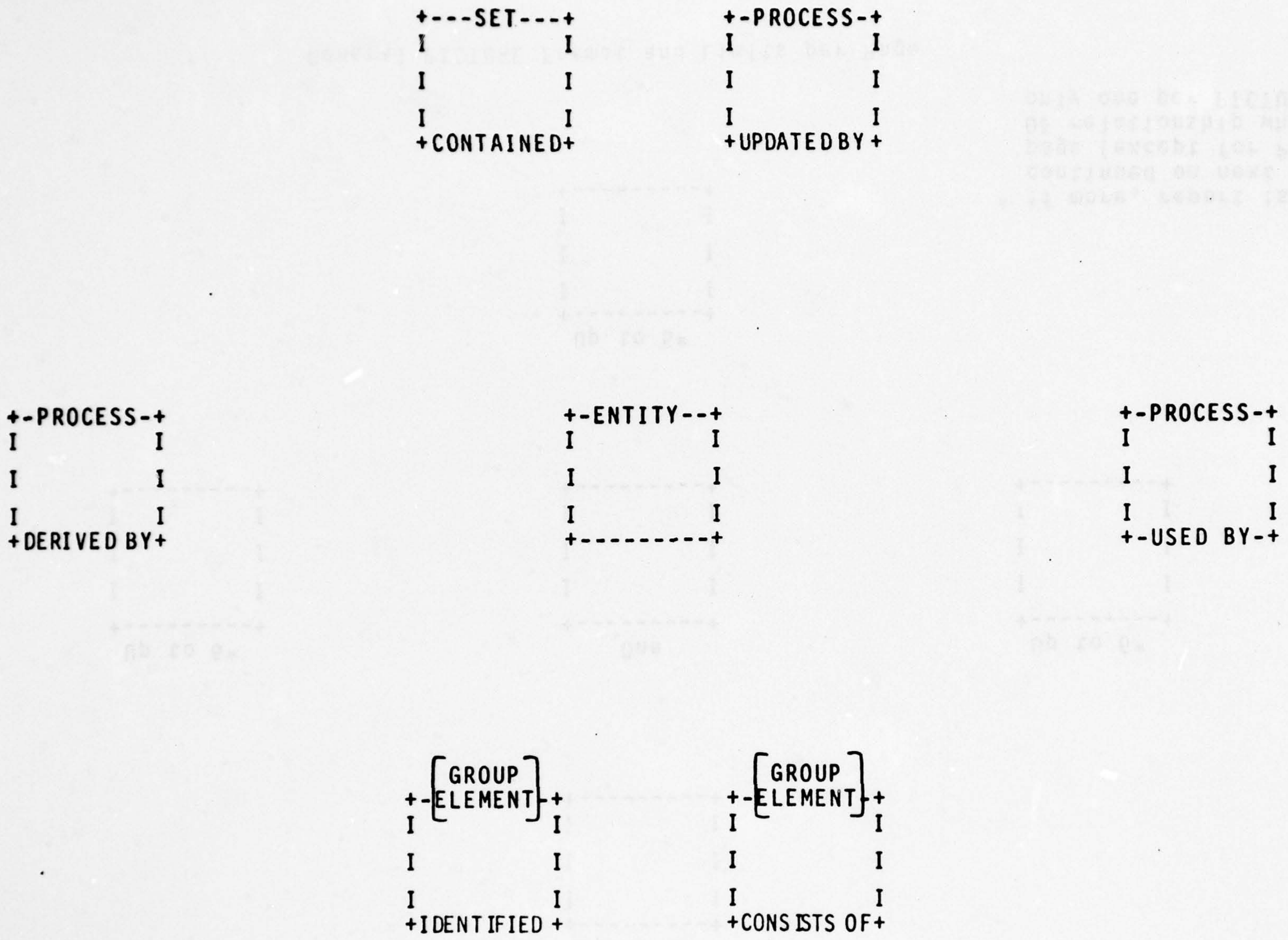
Analysis: The PICTURE REPORT involves a simple retrieval procedure similar to that in the FPS. The main difference is the manner of presentation. More formatting problems are encountered when implementing graphical, rather than narrative, descriptions for objects.

NOTE: The structure of PICTURE reports is shown on page 81a and the contents of PICTURE reports for each type of objects are shown on pages 81b-81h.

```
JRA VERSION 740710                                          JUL 11, 1974  21:38.52
                                 ADS-EXAMPLE
                                   PICTURE
PARAMETERS FOR: PIC
  NAME=PAYROLL-PROCESSING NOINDEX DATA NOSTRUCTURE FLOW
```

```
URA VERSION 740710                                         JUL 11, 1974  21:38.52
                                  ADS-EXAMPLE

                             PROCESS PICTURE

    PAYROLL-PROCESSING




+---INPUT---+                                                                         +--OUTPUT---+
I           I                                                                         IPAY-       I
I TIME-CARD I                                                                         ISTATEMENT  I
I           I   .                                                                  .  I           I
+USES TO DRV+      .                                                            .     +--DERIVES--+
                      .                                                      .
+---INPUT---+            .                                                .           +--OUTPUT---+
ITERMINATIN-I               .                                          .              IERROR-     I
IG-EMPLOYEE-I                  .                                    .                 ILISTING    I
I-INFO      I     .               .        +--PROCESS--+         .               .    I           I
+USES TO DRV+          .             .     IPAYROLL-   I      .             .         +--DERIVES--+
               ...............IPROCESSING I................
+---INPUT---+          .             .     I           I      .             .         +--OUTPUT---+
INEW-       I     .               .        +-----------+         .               .    IHIRED-     I
IEMPLOYEE-  I                  .                 .                  .                 ITERMINATED-I
IINFORMATIONI               .                    .                     .              IREPORT     I
+USES TO DRV+            .                       .                        .           +--DERIVES--+
                      .                          .                           .
+---INPUT---+      .                             .                              .     +--OUTPUT---+
IWEEKLY-    I   .                                .                                 .  IPAYSYSTEM- I
IEMPLOYEE-  I                                   . .                                   IOUTPUTS    I
IINFORMATIONI                                  .   .                                  I           I
+-RECEIVES--+                                 .     .                                 +-GENERATES-+
                                             .       .
                                            .         .
                                           .           .
                                    +--ENTITY---+ +----SET----+
                                    IVARYING-   I IPAYROLL-   I
                                    IEMPLOYEE-  I IMASTER-    I
                                    IDATA       I IINFORMATIONI
                                    +--UPDATES--+ +--UPDATES--+
                              Up to 5*
                             +---------+
                             I         I
                             I         I
                             I         I
                             +---------+


   Up to 6*                    One                          Up to 6*
  +---------+                +---------+                  +---------+
  I         I                I         I                  I         I
  I         I                I         I                  I         I
  I         I                I         I                  I         I
  +---------+                +---------+                  +---------+


                              Up to 5*
                             +---------+
                             I         I
                             I         I
                             I         I                * if more, report is
                             +---------+                  continued on next
                                                          page (except for PART
                                                          OF relationship which
                                                          only one per PICTURE).
```

General PICTURE Format and Limits per Page



ENTITY PICTURE

```
                                   +---RWE---+
                                   I         I
                                   I         I
                                   I         I
                                   +-PART OF-+




+-OUTPUT--+                        +---RWE---+                          +--INPUT--+
I         I                        I         I                          I         I
I         I                        I         I                          I         I
I         I                        I         I                          I         I
+-RECEIVES+                        +---------+                          +GENERATES+




                                   +---SET---+   +---RWE---+
                                   I         I   I         I
                                   I         I   I         I
                                   I         I   I         I
                                   +RESPONSIBLE +   +SUBPART IS+
```

RWE PICTURE

```
              +---SET---+               +--INPUT--+
              I         I               I         I
              I         I               I         I
              I         I               I         I
              +CONTAINED+               +-PART OF-+


                                                                  +-PROCESS-+
                                                                  I         I
                                                                  I         I
                                                                  I         I
+---RWE---+              +--INPUT--+                              +RECEIVED BY+
I         I              I         I
I         I              I         I
I         I              I         I
+GENERATED+              +---------+                              +-PROCESS-+
                                                                  I         I
                                                                  I         I
                                                                  I         I
                                                                  +-USED BY-+

               [GROUP  ]
             +-[ELEMENT]-+              +--INPUT--+
             I           I              I         I
             I           I              I         I
             I           I              I         I
             +CONSISTS OF+              +SUBPART IS+
```

INPUT PICTURE

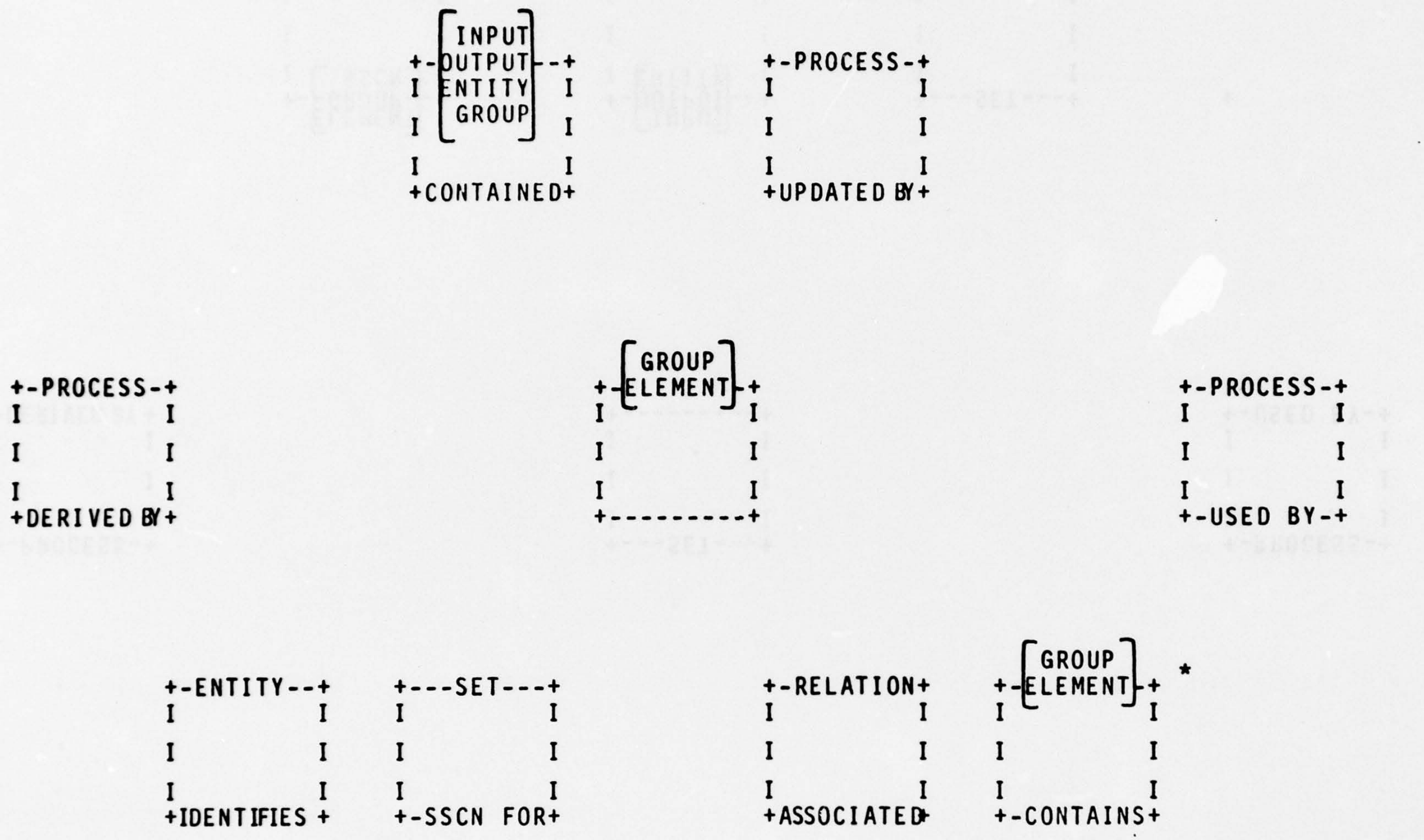
```
                 +---RWE---+        +---SET---+        +-PROCESS-+
                 I         I        I         I        I         I
                 I         I        I         I        I         I
                 I         I        I         I        I         I
                 +-RRWE IS-+        +SUBSET OF+        +UPDATED BY+

+-PROCESS-+                         +---SET---+                          +-PROCESS-+
I         I                         I         I                          I         I
I         I                         I         I                          I         I
I         I                         I         I                          I         I
+DERIVED BY+                        +---------+                          +-USED BY-+

                   ELEMENT              INPUT
                 +- GROUP --+        +-OUTPUT--+       +---SET---+        +
                 I  SSCN    I        I ENTITY  I       I         I
                 I          I        I         I       I         I
                 I          I        I         I       I         I
                 +-SSCA IS-+        +CONSISTS OF+      +SUBSET OF+
```

SET PICTURE

AD-A041 825 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
USER REQUIREMENTS ANALYZER VERSION 2.0 USERS MANUAL FOR IBM 370--ETC(U)
APR 75 C R MOORE, H J EIDEN
UNCLASSIFIED ESD-TR-75-88-VOL-3 NL

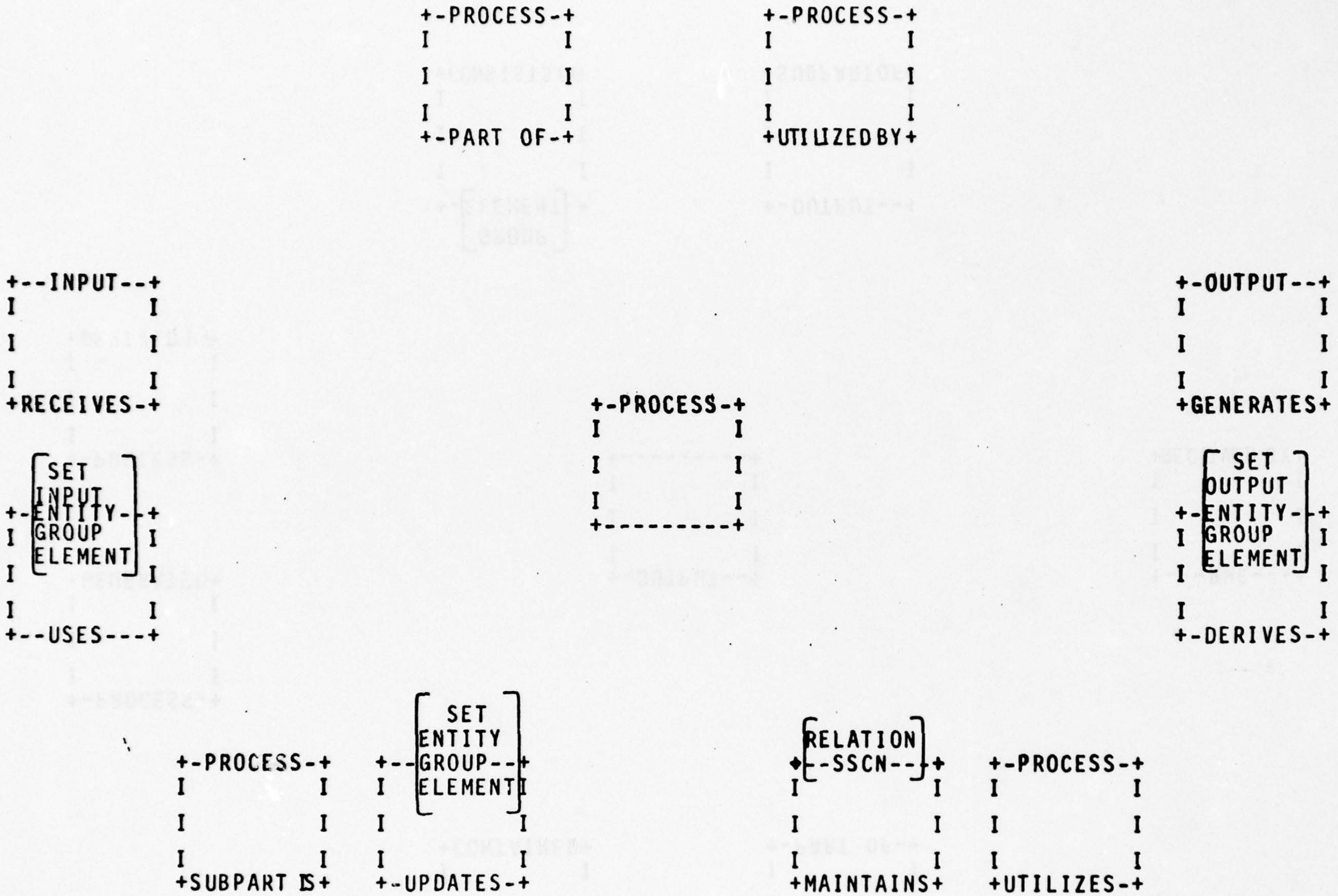
5 OF 5
AD
A 041825

END
DATE
FILMED
8-77



GROUP/ELEMENT PICTURE

* Pertains to GROUP PICTURE only



PROCESS PICTURE

```
                         +---SET---+            +-OUTPUT--+
                         I         I            I         I
                         I         I            I         I
                         I         I            I         I
                         +CONTAINED+            +-PART OF-+

+-PROCESS-+
I         I
I         I
I         I
+GENERATED+                          +-OUTPUT--+                            +---RWE---+
                                     I         I                            I         I
                                     I         I                            I         I
                                     I         I                            I         I
+-PROCESS-+                          +---------+                            +RECEIVED BY+
I         I
I         I
I         I
+DERIVED BY+

                           GROUP
                         +-ELEMENT-+            +-OUTPUT--+
                         I         I            I         I
                         I         I            I         I
                         I         I            I         I
                         +CONSISTSOF+           +SUBPARTOF+
```

OUTPUT PICTURE

11.2 DATA PROCESS REPORT

Purpose: To present both data/process relationships and completeness and consistency analysis for these relationships. This information is very valuable to the analyst for improving the system description and aids in specifying what information is needed to complete this description. This report can also be passed down to physical system designers to aid them in their work of improving the flow of data between processes and aids to design data organization.

Command: This report is generated whenever the DATA-PROCESS command is given with either the DATA parameter or the PROCESS parameter.

Options: If the DATA parameter is specified, the names used as input must be INPUT, OUTPUT, SET, ENTITY, GROUP or ELEMENT names. These names are then assigned numbers corresponding to the rows of the matrix and the PROCESSES which interact with this data are represented by the columns.

If the PROCESS parameter is specified, the names used as input must be PROCESS names. These names are then assigned numbers corresponding to the columns of the matrix, and the data which interact with these PROCESSES are represented by the columns.

Any relationships between PROCESS and DATA (column and row) is designated by an "I" (used as input to the process), a "U" (updated by the PROCESS) or "O" (is output from the PROCESS).

Contents and Order: The first part of the report is merely an index into the matrix specifying the names which represent a particular row or column number. The name types for each name are also printed. The second part is the DATA PROCESS INTERACTION MATRIX which actually shows the relationships between data and PROCESSES in matrix format. The summary section for this part is called DATA PROCESS INTERACTION MATRIX ANALYSIS and presents completeness information concerning the names used in the matrix.

The second matrix presented in this output designates those PROCESSES which derive or update data used by other PROCESSES. This relationship is designated by an asterisk. The data flow is from the PROCESSES represented by rows to the PROCESSES represented by columns. The summary for this matrix specifies those PROCESSES which have no interaction with other PROCESSES, have no immediate successors, and those with no immediate predecessors.

Analysis: Aside from the formatting procedures (printing the name lists and matrices) the most important analysis is done to produce the DATA PROCESS INTERACTION MATRIX ANALYSIS and the PROCESS INTERACTION MATRIX ANALYSIS. For the DATA PROCESS INTERACTION MATRIX ANALYSIS the PROCESS names are checked to see that they interact with data, use data if data is generated from the PROCESS, etc. the data in this analysis is checked to ascertain that it is derived by some PROCESS, received by some PROCESS (if it is an INPUT), etc. Special analysis is also done for the PROCESS INFORMATION MATRIX ANALYSIS. It specifies those PROCESSES which begin, end or are not involved in the flow of data for the system.

```
URA VERSION 740327                                             JUL  6, 1974   16:52.25
                                   ADS-EXAMPLE
                                     DATA PROCESS REPORT

PARAMETERS FOR: DP

  FILE PROCESS

  THE ROWS ARE DATA NAMES, THE COLUMNS ARE PROCESS NAMES.

   **ROW**                                 **COLUMN**

  1 ERROR-LISTING                          1 ERROR-LISTING-PRODUCTION
    (OUTPUT)                                 (PROCESS)
  2 BAD-INPUT-DATA                         2 FIELD-CHECK-NEW
    (GROUP)                                  (PROCESS)
  3 NEW-EMPLOYEE-PART                      3 FIELD-CHECK-PAYCALC
    (GROUP)                                  (PROCESS)
  4 VALID-NEW-INFO                         4 FIELD-CHECK-TERM
    (GROUP)                                  (PROCESS)
  5 NEW-EMPLOYEE-INFORMATION               5 FILE-REFERENCING
    (INPUT)                                  (PROCESS)
  6 PAYROLL-MASTER-INFORMATION             6 NEW-EMPLOYEE-PRINTING
    (SET)                                    (PROCESS)
  7 EMPLOYEE-DATA                          7 NEW-EMPLOYEE-PROCESSING
    (*** UNKNOWN OR AMBIGUOUS ***)           (PROCESS)
  8 VARYING-EMPLOYEE-DATA                  8 NEW-EMPLOYEE-UPDATING
    (ENTITY)                                 (PROCESS)
  9 FIXED-EMPLOYEE-DATA                    9 NEW-INFO-VALIDATION
    (ENTITY)                                 (PROCESS)
 10 VALID-T-CARD                          10 PAYCALC-INPUT-VALIDATION
    (*** UNKNOWN OR AMBIGUOUS ***)           (PROCESS)
 11 TIME-CARD                             11 PAYCALC-UPDATING
    (INPUT)                                  (PROCESS)
 12 CHECK                                 12 PAYCHECK-PRINTING
    (GROUP)                                  (PROCESS)
 13 STUB                                  13 PAYROLL-PROCESSING
    (GROUP)                                  (PROCESS)
 14 PAY-STATEMENT                         14 PAYSTATEMENT-PRODUCTION
    (OUTPUT)                                 (PROCESS)
 15 TERMINATING-EMPLOYEE-INFO             15 TERM-INFO-VALIDATION
    (INPUT)                                  (PROCESS)
 16 HIRED-TERMINATED-REPORT               16 TERMINATING-EMP-PRINTING
    (OUTPUT)                                 (PROCESS)
```

```
URA VERSION 740328                                                        JUL  6, 1974  16:52.25
                                ADS-EXAMPLE

                                  DATA PROCESS REPORT

THE ROWS ARE DATA NAMES, THE COLUMNS ARE PROCESS NAMES.

17 WEEKLY-EMPLOYEE-INFORMATION                17 TERMINATING-EMP-PROCESSING
   (INPUT)                                       (PROCESS)
18 PAYSYSTEM-OUTPUTS                          18 TERMINATING-EMP-UPDATING
   (OUTPUT)                                      (PROCESS)
19 VALID-TERM-INFO
   (*** UNKNOWN OR AMBIGUOUS ***)
20 TERMINATED-EMPLOYEE-PART
   (GROUP)
```

```
JRA VERSION 740326                                           JUL  6, 1974   16:52.25
                                    ADS-EXAMPLE

                                         DATA PROCESS REPORT

DATA PROCESS INTERACTION MATRIX

(I,J) VALUE    MEANING
-----------    --------------------------

    I          ROW I IS INPUT TO COLUMN J
     U         ROW I IS UPDATED BY COLUMN J
      O        ROW I IS OUTPUT OF COLUMN J

                                            1    1   1   1   1   1    1   1   1
       1   2   3   4   5    6   7   8   9   0    1   2   3   4   5    6   7   8
   +--------------------+--------------------+--------------------+------------+
 1 I   O                I                    I           O        I            I
 2 I I                  I               U   OI                   OI            I
 3 I                    I   O   O            I                    I            I
 4 I                    I I       I     O    I                    I            I
 5 I                    I     I       I      I         I          I            I
   +--------------------+--------------------+--------------------+------------+
 6 I                    I      U   U         I  U       U   U     I      U   U I
 7 I                    I           O        I I                  I            I
 8 I                    I         I          I          U  I      I            I
 9 I                    I         I          I          U  I      I            I
10 I                    I                   OI I                  I            I
   +--------------------+--------------------+--------------------+------------+
11 I                    I                 I  I         I   I      I            I
12 I                    I                    I   O I         O    I            I
13 I                    I                    I   O I         O    I            I
14 I                    I                    I       O   O   O    I            I
15 I                    I                    I         I       I  I     I      I
   +--------------------+--------------------+--------------------+------------+
16 I                    I                    I           O        I            I
17 I                    I                    I         I          I            I
18 I                    I                    I           O        I            I
19 I                    I                    I                   OI I          I
20 I                    I                    I                    I   O   O    I
   +--------------------+--------------------+--------------------+------------+
URA VERSION 740328                                          JUL  6, 1974  16:52.25
                              ADS-EXAMPLE

                                   DATA PROCESS REPORT

DATA PROCESS INTERACTION ANALYSIS

PAYROLL-MASTER-INFORMATION      (ROW    6) NOT DERIVED BY ANY PROCESS
PAYROLL-MASTER-INFORMATION      (ROW    6) UPDATED, BUT NOT USED BY ANY PROCESS
VARYING-EMPLOYEE-DATA           (ROW    8) NOT DERIVED BY ANY PROCESS
FIXED-EMPLOYEE-DATA             (ROW    9) NOT DERIVED BY ANY PROCESS
HIRED-TERMINATED-REPORT         (ROW   16) NOT GENERATED BY ANY PROCESS
WEEKLY-EMPLOYEE-INFORMATION     (ROW   17) NOT USED BY ANY PROCESS
PAYSYSTEM-OUTPUTS               (ROW   18) NOT DERIVED BY ANY PROCESS

FIELD-CHECK-NEW                 (COLUMN   2) DOES NOT INTERACT WITH ANY DATA
FIELD-CHECK-PAYCALC             (COLUMN   3) DOES NOT INTERACT WITH ANY DATA
FIELD-CHECK-TERM                (COLUMN   4) DOES NOT INTERACT WITH ANY DATA
FILE-REFERENCING                (COLUMN   5) DOES NOT INTERACT WITH ANY DATA
TERMINATING-EMP-UPDATING        (COLUMN  18) UPDATES SOMETHING, BUT DOES NOT USE ANYTHING
```

```
     RA VERSION 7+0328                                                JUL  6, 1974  16:52.25
                                  ADS-EXAMPLE

                                  DATA PROCESS REPORT

PROCESS INTERACTION MATRIX (INCIDENCE)


THE ROWS AND COLUMNS ARE PROCESS NAMES FROM ABOVE,
  AN ASTERISK IN (I,J) MEANS THAT SOMETHING DERIVED
  OR UPDATED BY PROCESS I IS USED BY PROCESS J.



                                     1 11111111
                            1234567890 12345678
                           +----------+--------+
                         1 I          I        I
                         2 I          I        I
                         3 I          I        I
                         4 I          I        I
                         5 I          I        I
                           +----------+--------+
                         6 I          I        I
                         7 I          I        I
                         8 I          I*       I
                         9 I*    * *  I        I
                        10 I*         I*       I
                           +----------+--------+
                        11 I          I *      I
                        12 I          I        I
                        13 I       *  I   *    I
                        14 I          I *      I
                        15 I*         I     *  I
                           +----------+--------+
                        16 I          I        I
                        17 I          I        I
                        18 I          I        I
                           +----------+--------+
                                                          JUL  6, 1974  16:52.25
                          ADS-EXAMPLE

                           DATA PROCESS REPORT

PROCESS INTERACTION MATRIX ANALYSIS

ERROR-LISTING-PRODUCTION         (ROW/COL   1) NO SUCCESSESORS FOR THIS PROCESS
FIELD-CHECK-NEW                  (ROW/COL   2) NO INTERACTION WITH OTHER PROCESSES
FIELD-CHECK-PAYCALC              (ROW/COL   3) NO INTERACTION WITH OTHER PROCESSES
FIELD-CHECK-TERM                 (ROW/COL   4) NO INTERACTION WITH OTHER PROCESSES
FILE-REFERENCING                 (ROW/COL   5) NO INTERACTION WITH OTHER PROCESSES
NEW-EMPLOYEE-PRINTING            (ROW/COL   6) NO SUCCESSESORS FOR THIS PROCESS
NEW-EMPLOYEE-PROCESSING          (ROW/COL   7) NO INTERACTION WITH OTHER PROCESSES
NEW-INFO-VALIDATION              (ROW/COL   9) NO PREDESSESORS FOR THIS PROCESS
PAYCALC-INPUT-VALIDATION         (ROW/COL  10) NO PREDESSESORS FOR THIS PROCESS
PAYCHECK-PRINTING                (ROW/COL  12) NO SUCCESSESORS FOR THIS PROCESS
PAYROLL-PROCESSING               (ROW/COL  13) NO PREDESSESORS FOR THIS PROCESS
TERM-INFO-VALIDATION             (ROW/COL  15) NO PREDESSESORS FOR THIS PROCESS
TERMINATING-EMP-PRINTING         (ROW/COL  16) NO SUCCESSESORS FOR THIS PROCESS
TERMINATING-EMP-PROCESSING       (ROW/COL  17) NO INTERACTION WITH OTHER PROCESSES
TERMINATING-EMP-UPDATING         (ROW/COL  18) NO INTERACTION WITH OTHER PROCESSES
```

11.3 PROCESS INPUT/OUTPUT

Purpose: To present in narrative outline form flow information for PROCESSES. The format of this output is intended to be easy to read and could be used in communicating the description of PROCESS flow to others. This output produces the same level of flow information as given in the PICTURE output (for PROCESSES).

Command: This output is generated whenever the PROCESS-INPUT-OUTPUT command is used.

Options: The PROCESS DESCRIPTION can be omitted from the output when NODESCRIPTION is specified. Otherwise the DESCRIPTION comment entry associated with each PROCESS is printed. The PROCEDURE comment entry associated with each PROCESS can be included as part of the description when the PROCEDURE parameter is specified. The user can be selective in choosing to have either data used as input or data used as output, or both, included in the printout. When the report may be fairly large, it is advantageous to specify the INDEX parameter to generate an index into the report.

Contents and Order: For each PROCESS name used as input to the command, the DESCRIPTION comment-entry for that name is printed out. Along with this is all the information corresponding to the USES, UPDATES, RECEIVES, GENERATES, and DERIVES statement associated to that PROCESS. The data names in these statements are classified as either INPUTS or OUTPUTS to the PROCESS and formatted as such.

Analysis: Production of this output basically involves a retrieval operation to obtain all data needed. A check is made and a comment printed for PROCESSES which do not have any inputs or outputs.

```
                                                          JUL  7, 1974  18:44.24
                                   ADS-EXAMPLE

                                      PROCESS INPUT/OUTPUT

PARAMETERS FOR: PROCIO

 FILE INPUTS OUTPUTS DESCRIPTIONS PRINT NOPUNCH



  1*  PAYROLL-PROCESSING

                          THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
                          IN THE TARGET SYSTEM.  IT ACCEPTS AND PROCESSES
                          ALL INPUTS AND PRODUCES ALL OUTPUTS.

                          ***  INPUTS  ***

     1  WEEKLY-EMPLOYEE-INFORMATION                RECEIVED
     2  TIME-CARD                                  USED TO DERIVE

                          ***  OUTPUTS  ***

     1  PAYSYSTEM-OUTPUTS                          GENERATED
     2  PAY-STATEMENT                              DERIVED
     3  ERROR-LISTING                              DERIVED
     4  HIRED-TERMINATED-REPORT                    DERIVED
     5  VARYING-EMPLOYEE-DATA                      UPDATED
     6  FIXED-EMPLOYEE-DATA                        UPDATED
     7  PAYROLL-MASTER-INFORMATION                 UPDATED


  2*  NEW-EMPLOYEE-PROCESSING

                          THIS PROCESS HANDLES THE OCCURRENCE OF THE INPUT
                      CALLED NEW-EMPLOYEE-INFORMATION.

                          ***  INPUTS  ***

     1  NEW-EMPLOYEE-INFORMATION                   USED TO DERIVE

                          ***  OUTPUTS  ***
```

```
                                                            JUL  7, 1974  13:44.24
                                    ADS-EXAMPLE

                                     PROCESS INPUT/OUTPUT

  1  NEW-EMPLOYEE-PART                               DERIVED
  2  PAYROLL-MASTER-INFORMATION                      UPDATED
```

12. DYNAMIC ANALYSIS OUTPUTS

These outputs intend to present the state of the system over time. Where the previous sections describe outputs which only give a static representation of the system, these outputs give a dynamic representation.

More specifically, they present information associated with the EVENTS and INTERVALS used in the system description.

12.1 URA FREQUENCY REPORT

Purpose: To present frequency information pertaining to events, occurrence of inputs, occurrence of outputs and processing. All information associated with the HAPPENS statement is presented by this report. This information will be needed in later stages of the design process when estimating workloads and specifying physical characteristics of the system. It can be used by the user to check for completeness of such information over the whole system and to check for accuracy of the current information.

Command: The report is initiated whenever the FREQUENCY command is given.

Options: At present, no variations in the report are available.

Contents and Order: Each INTERVAL name in the data base which has a HAPPENS statement related to it is printed out. Each object related to this INTERVAL via the HAPPENS statement is then printed out with its corresponding name type (either EVENT, INPUT, OUTPUT or PROCESS) and the system parameter which specifies the number of times this object occurs in the INTERVAL. The INTERVALS are ordered alphabetically.

Analysis: Production of this report mainly involves a retrieval process to extract the information relating to the HAPPENS statement.

```
                                                      JUL  6, 1974  16:01.44
                              ADS-EXAMPLE

                   PSA   FREQUENCY   REPORT

INTERVAL :  CALENDAR-WEEK

    NAME                              TYPE         TIMES  HAPPENS
    OCCURRENCE-OF-BAD-INPUT           EVENT        NUMBER-OF-BAD-INPUTS
    TERMINATING-EMPLOYEE-INFO         INPUT        1
    NEW-EMPLOYEE-INFORMATION          INPUT        1
    TIME-CARD                         INPUT        1
    PAY-STATEMENT                     OUTPUT       NUMBER-OF-EMPLOYEES
```

12.2 EVENT/CONDITION REPORT

Purpose: To present information relating EVENTS and CONDITIONS used in the user requirement. Such information is normally captured by a truth table. This output gives the user the ability to check for completeness in dealing with CONDITIONS and accuracy in defining the resulting EVENTS given a value for a CONDITION.

Command: At present this output is not available on the URL/URA system.

Contents and Order: This output will probably present the information available by the BECOMING statements for CONDITIONS which can readily be presented in matrix or truth table format. A summary will probably be given to aid in resolving redundancies and/or incompleteness.

12.3 DYNAMIC ANALYSIS REPORT

Purpose: This report is intended to present complex relationships among EVENTS, INTERVALS and PROCESSES (perhaps INPUTS and OUTPUTS will eventually be included). The main objective is to show the dependency between PROCESSES and EVENTS in order to derive a sequencing order of when PROCESSES occur. This report will also present information concerning consistency of frequency of EVENTS related to frequency of PROCESSES. For example, either EVENT El or E2 TRIGGERS PROCESS Pl, the frequency associated to Pl should be equal to the sum of frequencies El and E2. When this is not the case some explanation shpuld be specified. This report is intended as an aid to the analyst in detecting incompleteness and inconsistencies in the user requirement related to dynamic analysis. This report can be included as part of the final specifications.

Command: At present, this report has not been implemented on the URL/URA system.

Contents and Order: A matrix will probably be used to relate PROCESSES and EVENTS. EVENTS will be represented by the columns of the matrix and PROCESSES by the rows. A blank row would designate that the PROCESS represented by that row is independent of any EVENTS specified in the matrix. A summary section would also be included to present statistical information about the matrix. Another section would interrelate PROCESSES and EVENTS with INTERVALS as specified by the HAPPENS statements.

13. SIZE AND VOLUME ANALYSIS OUTPUTS

These outputs present information pertaining to the usage of system parameters in the user requirement. This information is necessary in estimating data storage and processing requirements during the physical design phase of the system building process. The data needed to derive these outputs can be extracted from the following URL statements:

CARDINALITY

CONSISTS

HAPPENS

CONNECTIVITY

These outputs serve to define the usage of system parameters in the user requirement as well as estimate the magnitude of the system in terms of amount of data handled and amount of processing required.

13.1 SYSTEM PARAMETER ANALYSIS REPORT

Purpose: This report provides the user with a check of the context in which each system parameter was used throughout the user requirement. It is intended as a tool to check consistency of usage and of conventions in dealing with system parameters. For example, naming conventions may be used in assigning system parameter names which should be kept consistent throughout user requirement. Inconsistency might also occur when the same system parameter name (the name, N, is a good example) may be unknowingly used to represent two different values It is the goal of this output to present such information.

Command: At present, this output has not been implemented on the URL/URA system.

Contents and Order: For each system parameter name this output would present the contexts in which the system parameter has been used:

Structure size - (via CONSISTS statement)

Object size - (via CARDINALITY and CONNECTIVITY statements)

Frequency - (via HAPPENS statement)

13.2 SYSTEM PARAMETER SIZE REPORT

- Purpose: To relate SYSTEM PARAMETER names to their corresponding values. This aids the user in locating inconsistencies (a SYSTEM PARAMETER, FIVE, having a value of 50) and incompleteness (no assigned value to a particular SYSTEM PARAMETER name. After the user determines that the system parameter definitions are correct this report can become a part of the final specifications.

  Through user requirement only the names of the SYSTEM PARAMETERS need be used. At the appropriate time numerical values may be assigned to these names via the VALUES statement in the DEFINE section.

  Command: At present, this output is not available on the URL/URA system.

  Contents and Order: For each SYSTEM PARAMETER name specified in the output its associated numerical value or value range will be printed. The names with no corresponding values will be flagged.

13.3 SET SIZE REPORT

Purpose: To estimate the size of SETS used in the user requirement. (When a SET corresponds to a file, ENTITIES may be thought to correspond to records in the file.) The estimate is important in the early stages of system design as it directly influences the physical system design process. This report is intended as a part of the final specifications of the logical design phase.

The output will present the use of SYSTEM PARAMETERS in relation to the CARDINALITY statements in the SET and ENTITY sections and the CONSISTS statement in the SET section. Inconsistencies in the use of SYSTEM PARAMETERS for establishing SET size can be determined and evaluated.

Command: This output has not been implemented at the present time.

Contents and Order: For each SET specified in the output its CARDINALITY statement and CONSISTS statements would be printed. The CARDINALITY statements associated with these ENTITIES used in the CONSISTS statements would also be presented. By inspection of the SYSTEM PARAMETER values associated with these statements, it would be trivial to produce a numerical size estimate (in terms of ENTITIES) of the SETS and determine inconsistencies and incompleteness through the assignment of these values.

13.4 PROCESSING VOLUME REQUIREMENTS REPORT

Purpose: To provide some estimate of the magnitude of the processing workload that must be taken into account by the physical system design. This report will become a part of the final specifications to be passed on to the physical system designer. The volume of processing can be estimated by the number of times EVENTS which trigger PROCESSES, and PROCESSES themselves, occur within a given time INTERVAL. Such information is easily extracted from the HAPPENS statements in the EVENT and PROCESS sections.

Command: This output is not currently available on the URL/URA system.

Contents and Order: For each PROCESS specified in the output, any HAPPENS statements related to that PROCESS will be printed as well as any HAPPENS statements related to these EVENTS which TRIGGER the PROCESS. Through this information the total number of times a PROCESS occurs for any given INTERVAL can be extracted. Where data is lacking to present these statistics, a warning will be given to that effect.

14. SYSTEM PROPERTY OUTPUTS

- These outputs specify aspects of the entire system not presented by the previous reports. In most cases this will mean performing analysis on the usage of ATTRIBUTES throughout the user requirement. By the assignment of particular ATTRIBUTE names, analysis can be performed on the values assigned these ATTRIBUTES names to extract information similar to the way in which the previous outputs extract information from particular URL statements.

14.1 ATTRIBUTE REPORT

Purpose: To present the values assigned to ATTRIBUTE names throughout the user requirement so that their usage may be checked for consistency and accuracy. For example, given an ATTRIBUTE, PROCESSING-MODE, the two values for it, ON-LINE and INTERACTIVE, may be synonymous in this application and so the terminology needs to be made consistent. This output may be used as a tool for the analyst in locating inconsistencies, etc., or may also be used as a quick reference in locating all objects that are associated with a particular ATTRIBUTE/ATTRIBUTE-VALUE pair.

Command: Production of this report is initiated through the PRINT-ATTRIBUTE-VALUES command.

Options: Currently no options are available that change content or format for this output.

Contents and Order: For each ATTRIBUTE specified in the report all the objects it is related to in the user requirement as well as the specific value the ATTRIBUTE takes on for that object.

Analysis: The process of implementing this report involves a basic retrieval and formatting procedure.

```
                                                       JUL  6, 1974  15:01.44
                                  ADS-EXAMPLE

                                     ATTRIBUTE REPORT

PARAMETERS FOR: PAY

   FILE


    1#  ATTRIBUTE: TYPE


              APPLIES TO:                                VALUE:
            1 OCCURRENCE-OF-BAD-INPUT                    RANDOM-EVENT
            2 NUMBER-OF-DEPENDENTS                       NUMERIC-INFORMATION
            3 PAYRATE                                    NUMERIC-INFORMATION
            4 EMPLOYEE-NAME                              CHARACTER-INFORMATION
            5 COMPLETE-PAY-INFORMATION                   MAINTAINED-WEEKLY
            6 TAX-RATE                                   NUMERIC-INFORMATION
            7 YTD-DEDUCT                                 NUMERIC-INFORMATION
            8 YTD-GROSS                                  NUMERIC-INFORMATION
            9 GROSS-PAY                                  NUMERIC-INFORMATION
           10 NET-PAY                                    NUMERIC-INFORMATION
           11 TERMINATION-CODE                           NUMERIC-INFORMATION
           12 PAYSTATEMENT-PRODUCTION                    WEEKLY-PROCESS
           13 TERMINATING-EMPLOYEE-INFO                  RECURRING-WEEKLY-INPUT
           14 NEW-EMPLOYEE-INFORMATION                   RECURRING-WEEKLY-INPUT
           15 TIME-CARD                                  RECURRING-WEEKLY-INPUT
           16 TERMINATING-EMP-PROCESSING                 RANDOM-PROCESS
           17 NEW-EMPLOYEE-PROCESSING                    RANDOM-PROCESS
           18 PAYROLL-MASTER-INFORMATION                 IMS-FORMAT
           19 ERROR-LISTING-PRODUCTION                   RANDOM-PROCESS
           20 HIRED-TERMINATED-REPORT                    RANDOM-OUTPUT
           21 ERROR-LISTING                              RANDOM-OUTPUT
           22 PAY-STATEMENT                              RECURRING-OUTPUT
```

14.2 COST/EFFECTIVENESS REPORTS

Purpose: To evaluate the cost/effectiveness of the proposed system (in the form of a user requirement) at any point in the user requirement definition process. These outputs would provide management and analyst the opportunity of "experimenting" with the user requirement to find the optimal cost/effectiveness. For example, if at one point in the system description it was suggested to add more functional capabilities at higher cost the use of the COST/EFFECTIVENESS REPORTS would present whether or not this expansion were feasible.

Command: This series of reports is not currently available on the URL/URA system.

Contents and Order: This information is intended to be contained in a package of reports all relating to various aspects of the cost/effectiveness problem. A separate paper will fully outline the contents of each report that makes up this package.

15. PROBLEM ANALYSIS OUTPUTS

All these outputs are generated in an attempt to improve the quality of the problem statement at any point in time. It is the task of these outputs to perform complex, special analysis to locate ambiguitites, inconsistencies, and incompleteness in the problem statement. All objects in the problem statement are checked for certain criteria particular to the contect in which they are used. For example, INPUTS and OUTPUTS should be related to RWES and PROCESSES for their description to be complete.

15.1 COMPLETENESS/CONSISTENCY REPORT

Purpose: To check that all necessary description has been specified for all objects in the user requirement at the point where it is acceptable as final specifications of the logical design phase. This is intended as an aid to the user to locate incomplete description of objects.

Command: This report is not currently available on the URL/URA system.

Contents and Order: The report will consist of a series of warnings specifying a particular object that has an incomplete description. Note that these are "warnings" and not "errors"; no statements are mandatory in the usage of URL/URA. A few of the things that must be checked are:

- Data such as ELEMENTS, GROUPS, ENTITIES, SETS, INPUTS and OUTPUTS interact in one way or another with one or more PROCESSES.
- EVENTS are related to INPUTS, OUTPUTS or PROCESSES.
- SYSTEM PARAMETERS and ATTRIBUTES have values associated with them.
- RWES interact with data (INPUTS or OUTPUTS)

Though this list is nowhere near exhaustive, it presents some of the requirements of forming a completeness/consistency report.

16. PROJECT MANAGEMENT OUTPUTS

These outputs are intended to be generated for and used by the project management as some record of progress in the use of URL/URA in the logical system design phase. The outputs present various statistics concerning the size of the user requirment based on the number of relationships specified in the data base and the types of relationships which have been specified. For example, it is possible to find out how many SYNONYMS have been used in the user requirement, etc. Each new URL statement added to the data base forms new relationships between objects and increases the size of the data base.

16.1 DATA BASE STATISTICS

Purpose: To present various statistics on the number of relationships specified in the data base. Progress of the project can be evaluated based on the number of names entered into the data base as well as the number of URL statements given for these names (i.e., the number of relationships specified). The output is intended to be run periodically to provide a hardcopy of the project progress for management.

Command: This output is generated by the DATA-BASE-STATISTICS command.

Options: Nubs represent the various statements in URL. It is possible, for each name used in the data base, to receive a count of the number of each type of nub associated with that name. In other words, it is possible to find out how many KEYWORDS a particular name has by looking at this output. To do this, the NAMNUBS parameter must be specified. SYNONYMS may be included in the list of names in the output but relatively little information is gained from this as SYNONYMS have no nubs directly associated with them.

Contents and Order: The first section consists of a list of all names in the URA data base. Associated with each name is first, a numeric code that signifies its name type (14 is equivalent to USER, for example) and then information, pertaining to the relationships the name is involved in. There are two possible relationships, RELA or RELB, to which a name can be associated. Within these relationships are various types of connections: NUBA, NUBB, NUBC, COM and OTHER. All numeric data beneath these headings correspond to counts for each type of connection within each type of relationship. TOT presents the total number of relationships specified for that name (or, RELA + RELB). The names from the data base are presented in alphabetical order.

The second part of the output presents statistics on the usage of each type of nub (or URL statement) in the URA data base. TYPE specifies a code for the type of nub and RELA and RELB present counts of the types of relationships specified for that type of nub. Note that if the nub represented a comment-entry statement only a count would be available for RELA and RELB would equal zero.

Analysis: Algorithms are used to maintain counts of relationships between names and URL statements.

```
                                                                    JUL  7, 1974  13:00.55
                                       ADS-EXAMPLE

                                       DATA BASE  STATISTICS

PARAMETERS FOR: DBS

   NAMES NUBS NOSYNONYMS NONAMNUBS

SEQ        NAME                           NUBA NUBB NUBC  COM  OTH RELA NUBA NUBB NUBC  COM  OTH RELB
  1 COMPANY-PROCEDURES-MANUAL       20     0    0    0    0    0    0    1    0    0    0    0    1
  2 DEPARTMENTS-AND-EMPLOYEES       16     0    0    0    1    0    1    2    0    0    0    0    2
  5 HIGHEST-LEVEL-PROCESS           10     0    0    0    0    0    0    1    0    0    0    0    1
  7 JOSEPH-ISMITH                   14     0    0    0    1    0    1    1    0    0    0    0    1
  8 LEVEL-1                         10     0    0    0    0    0    0    1    0    0    0    0    1
 12 PAYROLL-MASTER-INFORMATION      19     0    1    0    1    0    2    1    0    0    0    0    1
 13 PAYROLL-PROCESSING              15     6    0    0    1    0    7    1    1    0    0    0    2
 14 PAYSYSTEM-OUTPUTS               13     1    0    0    1    0    2    2    0    0    0    0    2
 16 RM-223H-WEST-ENGINEERING-BLDG   11     2    0    0    0    0    2    0    0    0    0    0    0
 17 TARGET-SYSTEM                   10     0    0    0    0    0    0    1    0    0    0    0    1
 18 WALTER-J-RATAJ                  14     4    0    0    1    0    5    1    0    0    0    0    1
 19 WEEKLY-EMPLOYEE-INFORMATION      8     1    0    0    1    0    2    2    0    0    0    0    2

    *** TOTALS                            14    1    0    7    0   22   14    1    0    0    0   15
```

```
                                  ADS-EXAMPLE                        JUL  7, 1974  13:00.55

                                        DATA BASE STATISTICS

TYPE (RELA,RELB)     TYPE (RELA,RELB)     TYPE (RELA,RELB)     TYPE (RELA,RELB)     TYPE (RELA,RELB)
   1 (   0,    0)      11 (   0,    0)      21 (   0,    0)      31 (   1,    1)      41 (   0,    0)
   2 (   0,    0)      12 (   0,    0)      22 (   3,    3)      32 (   0,    0)      42 (   0,    0)
   3 (   0,    0)      13 (   0,    0)      23 (   2,    2)      33 (   0,    0)      43 (   0,    0)
   4 (   0,    0)      14 (   7,    0)      24 (   0,    0)      34 (   0,    0)      44 (   0,    0)
   5 (   0,    0)      15 (   0,    0)      25 (   0,    0)      35 (   1,    1)      45 (   0,    0)
   6 (   0,    0)      16 (   1,    1)      26 (   0,    0)      36 (   0,    0)      46 (   0,    0)
   7 (   0,    0)      17 (   0,    0)      27 (   1,    1)      37 (   0,    0)      47 (   0,    0)
   8 (   0,    0)      18 (   0,    0)      28 (   4,    4)      38 (   0,    0)      48 (   0,    0)
   9 (   0,    0)      19 (   2,    2)      29 (   0,    0)      39 (   0,    0)      49 (   0,    0)
  10 (   0,    0)      20 (   0,    0)      30 (   0,    0)      40 (   0,    0)      50 (   0,    0)
```

16.2 DATA BASE SUMMARY

Purpose: Presents statistical information concerning the use of each name type possible in the data base. This output can be generated periodically to measure the amount of data that has been input into the URA data base over any time interval. Some degree of completeness can be estimated also as this output presents the percentage of names which have DESCRIPTIONS included in user requirement, the better the quality of the final specifications.

Command: This output is produced through usage of the SUMMARY command.

Options: No options are available for this command.

Contents and Order: All possible name types in the data base are listed in alphabetical order. For each name type certain statistics are presented. COUNT is the number of occurrences in the data base, of this particular name type. #W/SYN represents the number of these occurrences which have SYNONYMS assigned to them and PERCENT specifies this relationship in a percentage. #W/DESC represents the number of these occurrences which have DESCRIPTIONS associated with them and PERCENT specifies this relationship in a percentage.

Analysis: The generate of this output only requires the algorithms to perform the counts and then to format them.

```
                                                          JUL  6, 1974  16:01.44
                                  ADS-EXAMPLE

                                     DATA BASE SUMMARY

                                 COUNT #W/SYN PERCENT #W/DESC PERCENT
*** UNKNOWN OR AMBIGUOUS ***       5     2    40.00       0
ATTRIBUTE                          1     0                0
ATTRIBUTE-VALUE                   10     3    30.00       0
CONDITION                          1     0                0
ELEMENT                           13     8    61.53      10   76.92
ENTITY                             2     2   100.00       2  100.00
EVENT                              5     1    20.00       1   20.00
GROUP                              8     5    62.50       8  100.00
INPUT                              4     4   100.00       4  100.00
INTERVAL                           2     1    50.00       2  100.00
KEYWORD                            3     0                0
MAILBOX                            1     0                0
MEMO                               1     0                1  100.00
OUTPUT                             4     4   100.00       4  100.00
PROBLEM-DEFINER                    2     2   100.00       2  100.00
PROCESS                           18    13    72.22      12   66.66
REAL-WORLD-ENTITY                  3     3   100.00       3  100.00
RELATION                           1     1   100.00       1  100.00
SECURITY                           2     0                0
SET                                3     1    33.33       1   33.33
SOURCE                             1     0                0
SYSTEM-PARAMETER                   3     0                0

 ** TOTAL **                      93    50    53.76      51   54.83
```

GLOSSARY

analyst — Name used synonymously for "user." One who aids to develop the user requirement or logical system design.

Analyzer — Synonym for "URA." Is the software package that processes requirements stated in URL.

data base — Synonymous with "URA data base." This is the information sorted and retrieved by URA by means of the Modifier and Report commands.

FIFO — First in, first out

ISDOS Working Paper No. 100 — "Problem Statement Analyzer Report Generator Facilities." A steop-by-step guide to writing own reports for PSA.

logical description — Synonym to "user requirements." Set of requirements for a new system.

logical system design — The process of specifying a user requirement for any particular system.

name type — Any of the many types of names allowed by URL (i.e., PROCESS, SET, GROUP, etc.)

physical system design — The process of specifying a physical system (consisting of software, machinery, etc.) given a particular user requirement.

physical system designer — Person responsible for deriving a physical system design from the user requirement generated from the logical system design process.

user — Used synonymously with "analyst." That person who develops the requirements stated by the users into a format understandable by others and in sufficient detail to be usable by the physical system designer. The product of this work is the user requirement.

user requirements — A set of requirements specified by users of a proposed system and interpreted by the user into a format acceptable by the organization.

proposed system — Synonym for "target system." That system for which the user requirement is being developed.

URA — The User Requirements Analyzer. Synonym is "Analyzer." Software package which retrieves and inputs information to the URA data base.

URA data base — Area where URL information is stored (in a coded format) which can then be accessed by the commands allowed by URA.

URA "object" Synonymous with "name type." Any of the objects that can be defined by URL (a SET, a GROUP, a ELEMENT are all objects in URA).

URL statements Those statements specified by ISDOS Working Paper No. 68. The statement presents one aspect of description for a particular URA "object."

system description Synonymous with user requirements and logical description.

target system Synonymous with proposed system. The system that is intended to be the end result of logical and physical system design.

users That group of people who request a system or those people who specify the requirements for a system to be built.

Printed by
United States Air Force
Hanscom AFB, Mass. 01731